Generalised Acceptance Conditions for Symmetric Difference Nondeterministic Finite Automata

by

Laurette Marais

Dissertation presented for the degree of PhD in Computer Science in the Faculty of Science at Stellenbosch University

Supervisor: Prof. Lynette van Zijl

March 2018

Declaration

By submitting this dissertation electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof, save to the extent explicitly otherwise stated. In particular, Chapters 3 and 4 are based on three published papers, listed below, co-authored by me and my supervisor, Prof. Lynette van Zijl, of which my contribution comprises 90% of the content.

1. Marais, Laurette and Van Zijl, Lynette. Unary Self-Verifying Symmetric Difference Automata. In: Câmpeanu, C., Manea, F. and Shallit, J. (eds.) *Proceedings of the 18th International Conference on the Descriptional Complexity of Formal Systems, DCFS 2016, Bucharest, Romania, July 5-8, 2016*, pp. 180–191. LNCS, vol 9777. Springer, Cham, 2016.

2. Marais, Laurette and Van Zijl, Lynette. State Complexity of Unary SV-XNFA with Different Acceptance Conditions. In: Pighizzini G., Câmpeanu C. (eds.) *Proceedings of the 19th International Conference on the Descriptional Complexity of Formal Systems, DCFS 2017, Milan, Italy, July 3-5, 2017*, pp. 250–261. LNCS, vol 10316. Springer, Cham, 2017.

3. Marais, Laurette and Van Zijl, Lynette. Descriptional Complexity of Non-Unary Self-Verifying Symmetric Difference Automata. In: Erzsábet Csuhaj-Varjú, Pál Dömösi, György Vaszil (eds.) *Proceedings of the 15th International Conference on Automata and Formal Languages, AFL 2017, Debrechen, Hungary, September 4-6, 2017*, pp. 157–169. EPTCS, vol 252. 2017.

I declare that the reproduction and publication of this dissertation by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

March, 2018

Acknowledgements

In the beginning was the Word, and the Word was with God, and the Word was God. He was in the beginning with God. All things were made through him, and without him was not any thing made that was made.
John 1:1-2

This work would not exist aside from the ceaseless and sacrificial love of my husband, Willem, whom I can't thank enough. I am also very grateful to my 1-year-old daughter, Laurette, for being the loveliest girl a mother, especially one pursuing a PhD, could have.

I must thank my parents, who have believed in me for almost three decades and who provided me with their example of hard work and dedication: my father Fransjohan, and especially my mother, Laurette, also a computer scientist. I am privileged to have a mother who has been able to help me with my homework, from my first day in school to the very last days of my PhD studies.

My parents and my parents-in-law, Jaco and Elfrieda, also deserve special thanks for dedicating many hours to playing with their granddaughter while I disappeared into the study.

Many thanks are due to my supervisor, Prof. Lynette van Zijl, who always had an idea for what to try next when the results were not as expected, as so often happens; and to whom I could announce two pregnancies during the course of these studies and be greeted with unadulterated enthusiasm.

I would also like to thank Prof. Brink van der Merwe for providing some valuable answers and inputs along the way.

Finally, I am grateful to my colleagues at the HLT Research Group of the Meraka Institute, CSIR, and to Dr. Karen Calteaux in particular, whose determination to create a supportive working environment in any and all circumstances is most certainly unmatched.

Abstract

Generalised Acceptance Conditions for Symmetric Difference Nondeterministic Finite Automata

L. Marais

Dissertation: PhD (Comp. Sci.)

2018

Symmetric difference nondeterministic finite state automata (XNFA) are an instance of generalised nondeterminism, of which the behaviour is represented by the symmetric difference of all possible computation paths. We introduce the notion of *generalised acceptance* for XNFA, and investigate descriptional complexity issues related to two specific instances, namely self-verifying XNFA (SV-XNFA) and $\star$-XNFA.

For SV-XNFA, we apply self-verifying acceptance, originally defined for typical nondeterministic finite state automata (NFA), to XNFA. Self-verification involves defining a set of accept states, as well as a set of reject states, and requires that the automaton give an explicit accept or reject result on any input. We provide state complexity bounds for determinising unary and non-unary SV-XNFA.

We define $\star$-XNFA as XNFA with any finite number of final sets, while $\star$ represents a left-associative set operation on the language associated with each set of final states. We investigate and compare the descriptional complexity of various language operations, namely intersection, union, relative complement (or difference), symmetric difference and complement, for unary XNFA and unary $\star$-XNFA.

Uittreksel

Veralgemeende aanvaardingsvoorwaardes vir simmetriese-verskil nie-deterministiese eindige outomate

(Generalised Acceptance Conditions for Symmetric Difference Nondeterministic Finite Automata)

L. Marais

Proefskrif: PhD (Rek.wet.)

2018

Simmetriese verskil nie-deterministiese eindige outomate (XNFA) is 'n instansiëring van veralgemeende nie-determinisme, waarvan die gedrag gekenmerk word deur die simmetriese verskil van alle moontlike berekeningspaaie. Ons definieer *veralgemeende aanvaarding* vir XNFA en ondersoek aspekte van die beskrywingskompleksiteit van twee spesifieke instansiërings daarvan, naamlik self-verifiërende XNFA (SV-XNFA) en $\star$-XNFA.

Vir SV-XNFA pas ons self-verifiëring, wat oorspronklik vir tipiese nie-deterministiese eindige outomate (NFA) gedefinieer is, op XNFA toe. Self-verifiëring behels dat 'n versameling aanvaartoestande sowel as 'n versameling verwerpstoestande gedefinieer word, terwyl die vereiste is dat die outomaat enige invoer eksplisiet aanvaar of verwerp. Ons gee toestandskompleksiteitsgrense vir die determinering van unêre en nie-unêre SV-XNFA.

Ons definieer $\star$-XNFA as XNFA met enige eindige aantal versamelings finale toestande, terwyl $\star$ enige links-assosiatiewe versamelingsoperasie op die tale wat met die verskeie versamelings aanvaartoestande verband hou, voorstel. Ons ondersoek en vergelyk die beskrywingskompleksiteit van verskeie taaloperasies, naamlik snyding, vereniging, relatiewe komplement (of verskil), simmetriese verskil en komplement, vir unêre XNFA en unêre $\star$-XNFA.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Symmetric difference nondeterministic finite state automata (XNFA), first introduced in [31], are an instance of generalised nondeterminism that typically exhibit cyclic behaviour. This is due to the fact that the behaviour of an NFA is represented by the union of all its possible computation paths, while the behaviour of an XNFA is represented by the symmetric difference of all its possible computation paths. This difference is evidenced when applying the subset construction in order to find equivalent DFA, but it also has bearing on their respective definitions of acceptance. XNFA, specifically, exhibit so-called *parity acceptance*, which states that input is accepted if an odd number of paths lead to final states on a given input.

Alternative acceptance conditions, such as self-verification and various acceptance conditions associated with ω-automata, have been studied extensively for typical NFA [2; 6; 12; 13; 14; 23; 25], but have not been fully investigated for XNFA. A study of various acceptance conditions, which amounts to the addition or removal of certain constraints upon XNFA, would contribute to a fuller understanding of various aspects of the cyclic behaviour of XNFA.

Contrary to typical NFA, XNFA are most effectively studied with reference to the characteristic polynomials associated with each XNFA. These polynomials are found by considering the so-called transition matrices of XNFA, i.e., matrices that encode the transitions on each alphabet symbol of the XNFA. Such matrices consist of ones and zeroes, and in keeping with the symmetric difference character of XNFA, their characteristic polynomials are defined over the finite field of two elements, namely $GF(2)$, where addition is defined as XOR.

We therefore approach the question of studying alternative acceptance conditions for XNFA from a descriptional complexity point of view, via the mechanism of considering the characteristic polynomials associated with XNFA.

Self-verifying NFA (SV-NFA), which require that an automaton give an explicit, "trustworthy" result on any input, have two sets of final states, namely *accept* states

and *reject* states. We investigate how self-verification might be applied to XNFA, especially in light of parity acceptance, resulting in so-called self-verifying XNFA (SV-XNFA).

We consider self-verification as an instance of generalised acceptance, which allows multiple sets of final states, and we investigate another such instance, namely $\star$-XNFA. The difference between SV-XNFA and $\star$-XNFA is found in the nature of the constraints and meaning associated with multiple sets of final states. In the case of SV-XNFA, the two sets are associated with acceptance and rejection, and must be chosen so that the automaton provides an explicit *accept* or *reject* result on any input. In the case of $\star$-XNFA, any finite number of sets of final states is possible, and $\star$ is a left-associative set operation applied to the result associated with each set of final states.

We establish state complexity bounds on determinising SV-XNFA, based on the insight that a choice of accept and reject states which results in a succinct SV-XNFA is possible only if $X+1$ is a factor of the polynomials associated with a given XNFA.

$\star$-XNFA provide a natural context within which to study various set operations on languages represented by XNFA. We establish some descriptional complexity bounds for the operations of intersection, union, relative complement (or difference), symmetric difference and complement for unary XNFA and $\star$-XNFA, showing that in the case of intersection, union and relative complement, there exists a gap between $\star$-XNFA and XNFA, which we conjecture to be at least polynomial. However, for symmetric difference and complement, the bounds for $\star$-XNFA and XNFA are equal. Therefore, while our results show that $\star$-XNFA are suitable for representing some languages more succinctly than XNFA, they also serve to highlight the characteristic properties of XNFA that enable them to succinctly represent certain kinds of languages.

We find, therefore, that the notion of generalised acceptance is a valuable tool for investigating the behaviour of XNFA in general, as well as a means of providing more succinct representations of certain languages than what is possible with XNFA.

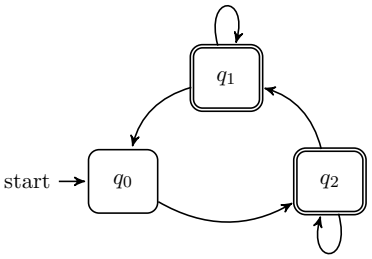
Chapter 2

Background

Symmetric difference nondeterministic finite state automata (XNFA) were first introduced as instances of generalised nondeterminism by Van Zijl in [31]. Typical nondeterministic finite state automata (NFA) are characterised by the union set operation, in the sense that the equivalent deterministic finite automaton (DFA) is found through the subset construction [24] by taking the union of all possible nondeterministic states in which the NFA can be at any given point. In other words, when considering the computation tree of the NFA when reading an input word, the union of all the states at the same depth in the tree is used to form the equivalent DFA state (see Example 1, page 11). Generalised nondeterminism generalises this aspect of the behaviour of finite state automata, by allowing any binary symmetric set operation to be used in the subset construction. In particular, XNFA are the instance where the behaviour of the automaton is characterised by the symmetric difference set operation.

We give a small comparative example: consider the unary automaton given in Figure 2.1. Its initial state is q_0 and its final states are q_1 and q_2. If we consider it as an NFA, Figure 2.2 shows its computation tree on input a^5. There are 12 leaf nodes, and hence there are 12 paths through the automaton that could possibly lead to acceptance on input a^5. As it happens, nine of these paths do lead to accept states. On the other hand, if we consider the automaton as an XNFA, Figure 2.3 shows its computation tree on the same input a^5. Here, after reading a^3, the characteristic symmetric difference behaviour of XNFA becomes apparent. Only input for which an odd number of paths exist can be accepted, and hence, at each level of the tree, any state with an even number of paths leading to it can be pruned from the tree, as indicated in grey. This leads to a much slimmer tree, with only two leaf nodes and hence only two paths through the automaton that could possibly lead to acceptance on input a^5. We see that exactly one path leads to an accept state.

An important difference between NFA and XNFA relates to their acceptance



δ	a
q_0	$\{q_2\}$
q_1	$\{q_0, q_1\}$
q_2	$\{q_1, q_2\}$

Figure 2.1: (X)NFA with transition table

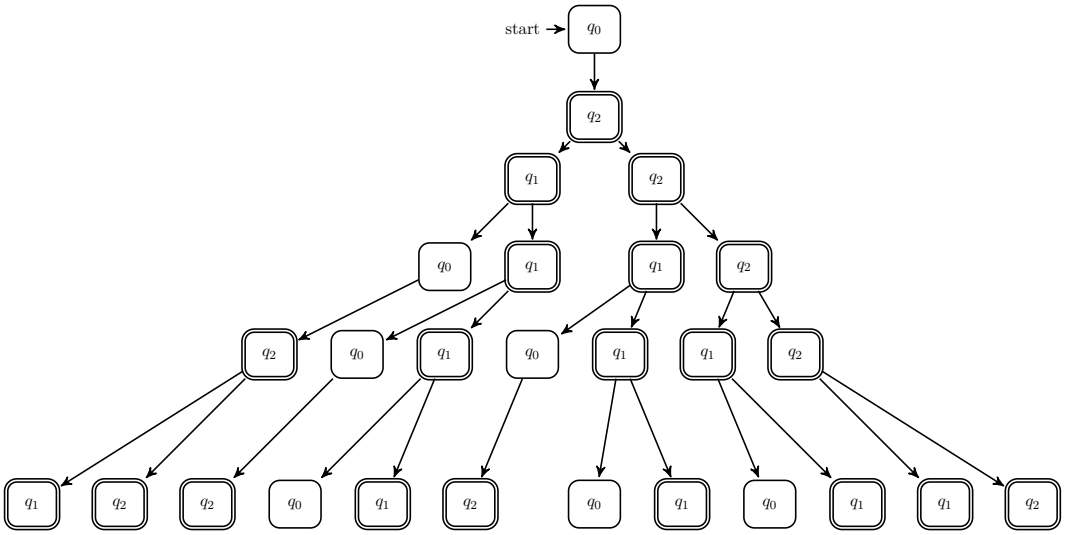


Figure 2.2: Computation tree of NFA

conditions: an NFA accepts a word w if a single accept state is among any of the states reached on w, while an XNFA accepts a word w if among the states reached on w, an odd number are accept states. This is known as parity acceptance [33], and reflects the parity character of the symmetric difference operation on sets – an element is in the symmetric difference of two sets if it belongs to either, but not both, sets. In the automaton sense, the NFA considers whether any path leads to a final state, while the XNFA considers whether there is an odd number of paths leading to a final state. In our example above, note that after reading a^2, both the NFA and the XNFA have two paths that reach final states. This means that the NFA accepts a^2, but since two is an even number, the XNFA rejects a^2.

Given parity acceptance, XNFA have been shown [28] to be equivalent to weighted automata over the finite field of two elements, namely $GF(2)$, and are known to typically have equivalent DFA that consist of cycles, which in the unary case are similar to the cycles of linear feedback shift registers (LFSRs) [33]. Hence, XNFA can, for example, be used for random number generation [33].

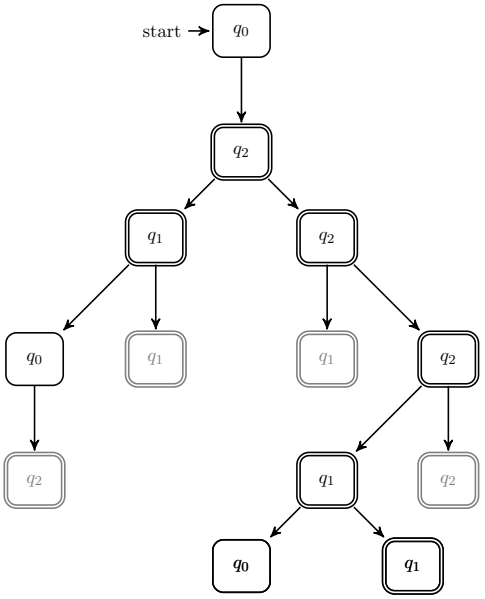


Figure 2.3: Computation tree of XNFA

XNFA are particularly interesting from a descriptional complexity point of view. For example, the tight upper bound on the number of states of a DFA equivalent to a given n-state unary XNFA is $2^n - 1$ [34], while the upper bound is $e^{\sqrt{n \log n}}$ for unary NFA [4]. Furthermore, Champarnaud et al. show in [3] that, compared to NFA, a larger number of regular languages can be succinctly represented by XNFA.

Other results related to descriptional complexity are work done on minimisation of XNFA for both the unary case by Van Zijl in [37], as well as for larger alphabets by Vuillemin and Gama in [39], and ambiguity [27; 30; 36]. It has also been shown that XNFA provide a normal form representation of languages, since $\mathcal{L} = \mathcal{L}' \Rightarrow \text{MXA}(\mathcal{L}) = \text{MXA}(\mathcal{L}')$, where $\text{MXA}(\mathcal{L})$ is the normalised minimal XNFA that accepts $\mathcal{L}$ [38]. The question of magic numbers for XNFA has also been studied [35].

The emphasis in previous work has been on various aspects of the descriptional complexity of XNFA, and often the results have been presented in comparison to results for NFA. One aspect that has not been fully investigated for XNFA is that of alternative acceptance conditions. For NFA, various kinds of acceptance conditions have been studied, including, for example, results for various kinds of ω-automata [2; 23; 25], as well as self-verifying acceptance [12]. In this work, we study certain alternative acceptance conditions in order to contribute to a more complete understanding of XNFA. Specifically, we define the notion of *generalised acceptance* for XNFA, focusing on two instances, namely self-verifying XNFA (SV-XNFA) in Chapter 4 and $\star$-XNFA, which we introduce in Chapter 5. The latter will be used

to shed some light on the descriptional complexity of operations on languages represented by XNFA. Note that $\star$-XNFA, introduced in this work as an instance of generalised *acceptance* for XNFA, should not be confused with $\star$-NFA, introduced in [31] as automata with generalised *nondeterminism*.

Self-verifying nondeterministic finite automata (SV-NFA) were originally introduced in the context of Las Vegas computation [6; 12; 13; 14], and involves the definition of two sets of final states for a single NFA, namely a set of *accept* states and a set of *reject* states. A Las Vegas computation on a self-verifying automaton searches through possible paths on a given input and stops when a path ending in either an accept state or a reject state is reached. Contrary to traditional NFA, where rejection is the result of failing to find a path that ends in an accept state, reaching a reject state in a self-verifying automaton confirms that the input word is not in the language accepted by the automaton. In fact, it is required that on any possible input, at least one path reaches either an accept state or a reject state, and that on no input two different paths exist that end in opposite kinds of final states. Hence, for any input word w, an SV-NFA returns an explicit accept or reject result.

The descriptional complexity of SV-NFA has been studied extensively, with Jiraskova and Pighizzini [16] providing a tight upper bound for non-unary alphabets as a function $g(n)$ on the number of NFA states, where $g(n)$ grows like $3^{\frac{n}{3}}$. For unary alphabets, Geffert and Pighizzini [7] give a lower bound of $e^{\Omega(\sqrt[3]{n \cdot \ln^2 n})}$. Furthermore, Jirásek et al. [15] prove tight complexity bounds on various non-unary operations on languages represented by SV-NFA, namely complement, intersection, union, relative complement (or difference), symmetric difference, reversal, star, left and right quotients, and an asymptotically tight bound on concatenation.

The descriptional complexity of Las Vegas automata has also been studied on so-called promise problems [8]. In [9], promise problems are described as problems that partition the set of all possible inputs into three subsets, namely strings representing YES-instances (or accepted strings), strings representing NO-instances (or rejected strings) and the set of disallowed strings (strings "promised" not to be presented to the automaton). In [8], Geffert shows that the gap between Las Vegas automata and DFA on promise problems is exponential. In [22], Moreira and Pighizzini define so-called Don't Care NFA and Don't Care DFA in order to consider a similar problem. Don't Care automata (dcNFA) have accepting and rejecting states, but there is no requirement that an explicit result be given on any possible input. The complexity of minimisation of Don't Care automata has been shown to be NP-complete in the deterministic case, and PSPACE-hard in the nondeterministic case [22].

In this work, we apply the concept of self-verification to XNFA. Self-verification imposes certain constraints on nondeterministic automata, and hence in applying

the notion to XNFA, we study the consequences of these constraints on XNFA. We therefore consider the descriptional complexity of self-verifying XNFA (SV-XNFA) in terms of deriving equivalent minimal deterministic automata. Since XNFA have parity acceptance, which requires counting the number of paths to final states, a straight forward application of self-verification to XNFA removes the relationship to Las Vegas computations that exists for SV-NFA. However, studying the descriptional complexity of SV-XNFA leads to a fuller understanding of the characteristic cyclical behaviour of XNFA.

Given that self-verification involves two sets of final states with specific meaning associated with each (namely accept and reject), along with specific constraints attached, we may reasonably ask in which ways this can be generalised. For example, dcNFA also have two sets of states that represent acceptance and rejection, but with fewer constraints. We therefore define generalised acceptance for XNFA as the notion that there may be any finite number of sets of final states, while any specific instance of generalised acceptance may determine the meaning that is associated with the sets of final states.

The remainder of this dissertation is structured as follows. In Chapter 3, we provide an overview of XNFA, highlighting the properties that are relevant to this study. We then define generalised acceptance for XNFA, and mention two specific instances, namely self-verifying XNFA and so-called $\star$-XNFA, with $\star$ representing any left-associative set operation.

In Chapter 4, we provide descriptional complexity results for determinising SV-XNFA, providing a tight bound for the unary case of $2^{n-1}-1$ and upper and lower bounds for larger alphabets, namely 2^n-1 and 2^{n-1} respectively. One of the questions discussed is the influence of parity acceptance on how self-verification is to be appropriately applied to XNFA. The original context for SV-NFA is that of Las Vegas computation, where a single instance of a path ending in a final state is confirmation that the input is accepted or rejected (depending on the kind of final state). This is similar to the fact that for NFA, a single instance of a path ending in an accept state is confirmation that the input is accepted. For XNFA, however, parity acceptance requires knowledge of the number of paths that end in accept states, and therefore similarly, SV-XNFA must in some way incorporate such knowledge. As we will see, two approaches seem reasonable, and we discuss the relative merits of both.

In Chapter 5, we define $\star$-XNFA as another instance of generalised acceptance in order to illustrate certain aspects of descriptional complexity bounds for operations on unary languages represented by XNFA. We provide upper and lower descriptional complexity bounds for the union, intersection, symmetric difference and relative

complement operations on unary languages represented by XNFA, before showing that in the cases of union, intersection and relative complement, unary $\star$-XNFA have improved bounds over those of XNFA. We conjecture that this gap is at least polynomial, and hence that $\star$-XNFA can be used to succinctly represent a larger number of unary languages than XNFA. In the case of the symmetric difference operation, the bound is identical for XNFA and $\star$-XNFA, and we discuss how this highlights certain useful properties of XNFA.

Finally, we conclude in Chapter 6 with a discussion of the results presented, as well as considering possible avenues for future research.

Chapter 3

Symmetric difference nondeterministic finite state automata

In this chapter we provide a brief introduction to symmetric difference nondeterministic finite state automata (XNFA). We first describe XNFA in terms of generalised nondeterminism, introduced in [31], and then present XNFA as weighted automata over the finite field of two elements, or $GF(2)$. We give an overview of the properties of unary and r-ary XNFA that lay the foundation for the work in Chapters 4 and 5, before introducing and defining the concept of generalised acceptance for XNFA, which forms the central theme of this work.

3.1 XNFA as instances of generalised nondeterminism

In this section, we show how typical NFA and XNFA are both instances of generalised nondeterminism, with NFA behaviour characterised by the union set operation, while XNFA behaviour is characterised by the symmetric difference set operation.

An NFA N is a five-tuple $N = (Q, \Sigma, \delta, Q_0, F)$, where Q is a finite set of states, Σ is a finite alphabet, $\delta : Q \times \Sigma \to 2^Q$ is a transition function (where 2^Q indicates the power set of Q), $Q_0 \subseteq Q$ is a set of initial states, and $F \subseteq Q$ is the set of final, or acceptance, states.

The transition function can be extended to $\delta : 2^Q \times \Sigma \to 2^Q$ in the following way:

$$\delta(A, \sigma) = \bigcup_{q \in A} \delta(q, \sigma) \ .$$

Furthermore, δ can be extended to strings in the Kleene closure Σ^* of the alphabet by

$$\delta(A, \epsilon) = A \text{ and}$$
$$\delta(A, wa) = \delta(\delta(A, w), a) \text{ for each } w \in \Sigma^* \text{ and each } a \in \Sigma.$$

An NFA N is said to accept a string $w \in \Sigma^*$ if and only if $\delta(Q_0, w) \cap F \neq \emptyset$, and the set of all strings (also called words) accepted by N is the language $\mathcal{L}(N)$ accepted by N. Any NFA has an equivalent DFA which accepts the same language. The DFA $N_D = (Q_D, \Sigma, \delta_D, Q_0, F_D)$ that is equivalent to a given NFA is found by performing the subset construction [11], so that Q_D consists of sets of states from Q. In essence, the subset construction keeps track of all the states that the NFA may be in at the same time, and forms the states of the equivalent DFA by a grouping of the states of the NFA. In short,

$$\delta_D(A, \sigma) = \bigcup_{q \in A} \delta(q, \sigma)$$

for any $A \subseteq Q$ and $\sigma \in \Sigma$. Any A is a final state in the DFA if $A \cap F \neq \emptyset$.

In terms of generalised nondeterminism [31], NFA are known as $\cup$-NFA, since the union set operation is used to define their behaviour.

A symmetric difference NFA (XNFA) is defined similarly to an NFA, except that their behaviour is defined by the symmetric difference set operation. For any two sets A and B, the symmetric difference is given by

$$A \oplus B = (A \cup B) \setminus (A \cap B) \ .$$

More specifically, an XNFA $N_\oplus$ is a five-tuple $N_\oplus = (Q, \Sigma, \delta, Q_0, F)$, with each element defined as for NFA with the exception of δ, which is extended to $\delta : 2^Q \times \Sigma \to 2^Q$ in the following way:

$$\delta(A, \sigma) = \bigoplus_{q \in A} \delta(q, \sigma) \ .$$

Just as for NFA, δ can be extended to strings in the Kleene closure Σ^* of the alphabet.

An XNFA $N_\oplus$ is said to accept a string $w \in \Sigma^*$ if and only if $|\delta(Q_0, w) \cap F|$ is odd, as an analogy to the symmetric difference set operation [38], also known as parity acceptance. In other words, an odd number of paths labeled w must lead to final states. The set of all strings (also called words) accepted by $N_\oplus$ is the language $\mathcal{L}(N_\oplus)$.

To determine the equivalent deterministic finite automaton, which we denote with XDFA for clarity, the subset construction is applied as

$$\delta_D(A,\sigma) = \bigoplus_{q\in A} \delta(q,\sigma)$$

for any $A \subseteq Q$ and $\sigma \in \Sigma$.

The XDFA is denoted with $N_{D,\oplus} = (Q_D, \Sigma, \delta_D, Q_0, F_D)$. An XDFA final state contains an odd number of final XNFA states. Since an equivalent deterministic finite automaton can be found for any given XNFA, XNFA accept the class of regular languages [38], and in terms of generalised nondeterminism are known as $\oplus$-NFA [31].

Example 1. *Let $N = (Q, \Sigma, \delta, Q_0, F)$ be some unary nondeterministic finite automaton, with $Q = \{q_0, q_1, q_2, q_3\}$, $\Sigma = \{a\}$, $Q_0 = F = \{q_0\}$ and δ as given below.*

δ	a
q_0	$\{q_1\}$
q_1	$\{q_2\}$
q_2	$\{q_3\}$
q_3	$\{q_0, q_2, q_3\}$

If we interpret N as a $\cup$-NFA, the transition function that results from applying the subset construction is δ_{DFA}, shown in Table 3.1[1]*. However, if we interpret N as a $\oplus$-NFA, the transition function that results from applying the subset construction is δ_{XDFA}, shown in Table 3.2. Let us compare the transitions from state $[q_0, q_2, q_3]$ in the two tables. In both, we have $\delta([q_0], a) = [q_1]$, $\delta([q_1], a) = [q_2]$, and $\delta([q_3], a) = [q_0, q_2, q_3]$. Therefore, in Table 3.1, $\delta([q_0, q_2, q_3], a) = [q_0, q_1, q_2, q_3]$, since $\{q_1\} \cup \{q_2\} \cup \{q_0, q_2, q_3\} = \{q_0, q_1, q_2, q_3\}$. On the other hand, in Table 3.2, $\delta([q_0, q_2, q_3], a) = [q_0, q_1, q_2]$, since $\{q_1\} \oplus \{q_2\} \oplus \{q_0, q_2, q_3\} = \{q_0, q_1, q_2\}$.*

Table 3.1: Transition function of N_{DFA}

	δ_{DFA}	a
$\rightarrow$	$[q_0]$	$[q_1]$
	$[q_1]$	$[q_2]$
	$[q_2]$	$[q_3]$
	$[q_3]$	$[q_0, q_2, q_3]$
$\leftarrow$	$[q_0, q_2, q_3]$	$[q_0, q_1, q_2, q_3]$
$\leftarrow$	$[q_0, q_1, q_2, q_3]$	$[q_0, q_1, q_2, q_3]$

Table 3.2: Transition function of N_{XDFA}

	δ_{XDFA}	a
$\rightarrow$	$[q_0]$	$[q_1]$
	$[q_1]$	$[q_2]$
	$[q_2]$	$[q_3]$
	$[q_3]$	$[q_0, q_2, q_3]$
$\leftarrow$	$[q_0, q_2, q_3]$	$[q_0, q_1, q_2]$
$\leftarrow$	$[q_0, q_1, q_2]$	$[q_1, q_2, q_3]$
	$[q_1, q_2, q_3]$	$[q_0]$

[1] For notational convenience, we indicate initial states with the arrow $\rightarrow$, and final states with the arrow $\leftarrow$.

□

For clarity, we indicate DFA and XDFA states with square brackets and use curly brackets when referring specifically to sets of states in the context of NFA and XNFA. However, we may still treat DFA and XDFA states as sets, by, for example, using the standard notation for indicating the membership of elements, i.e. $q_0 \in [q_0, q_1, q_2]$.

3.2 XNFA as weighted automata over $GF(2)$

XNFA have been shown in [29; 38] to be equivalent to weighted automata over the finite field (Galois field) of two elements, or $GF(2)$. The finite field $GF(2)$ consists of the elements 1 and 0. Multiplication is defined as usual, which means that it is equivalent to the boolean AND operation. Addition, however, is defined as shown in Table 3.3, and hence is equivalent to the boolean XOR operation. Specifically, in $GF(2)$, $1 + 1 = 0$. Note also that in $GF(2)$, $a - b = a + b$.

Table 3.3: Addition in $GF(2)$

+	0	1
0	0	1
1	1	0

Let $N = (Q, \Sigma, \delta, Q_0, F)$ be a unary XNFA with n states and $\Sigma = \{a\}$. We can represent the transition function $\delta : Q \times \Sigma \to 2^Q$ as an $n \times n$ matrix M over $GF(2)$ of which the (p, q)-th entry represents the weight (1 or 0) of the transition from p to q. Note that by assigning a weight of zero to non-transitions, the weighted automaton is defined to be complete, even if the XNFA is partial. The resulting matrix has a characteristic polynomial $c(X) = \det(XI - M)$, where I is the identity matrix. In this section we discuss the unary case, where a single matrix encodes all transitions. For larger alphabets, each letter in Σ is associated with a binary transition matrix as described. We discuss the case of r-ary XNFA more fully in Section 3.5.

Besides encoding δ in the transition matrix, we encode the initial states Q_0 as a vector of length n of elements in $GF(2)$, namely $v(Q_0) = [q_{0_0}\ q_{0_1}\ \cdots\ q_{0_{n-1}}]$, where $q_{0_i} = 1$ if $q_i \in Q_0$ and $q_{0_i} = 0$ otherwise. Similarly, we encode the final states as a vector of length n, namely $v(F) = [q_{F_0}\ q_{F_1}\ \cdots\ q_{F_{n-1}}]$. We abuse notation by letting $\delta : Q \times \Sigma \to 2^Q$ (a function to sets of states) and $\delta : Q \times \Sigma \to [GF(2)]^n$ (a function to vectors of length n over $GF(2)$) depending on the context. Then the weight of a word w_k of length k is given by

$$\Delta(w_k) = v(Q_0)M^k v(F)^T.$$

In fact, $v(Q_0)M^k$ is a vector that encodes the XNFA states reachable from the initial states after reading k letters, or equivalently, it encodes the XDFA state that is reached from the initial state after reading k letters. That is, $v(Q_0)M^k$ encodes $\delta(Q_0, w_k)$.

The important advantage of this interpretation is the fact that one can perform a so-called *change of basis* on the transition matrix and initial and final state vectors of an XNFA to produce an equivalent XNFA. This ability is essential in, for example, minimisation algorithms for XNFA [29].

Given an XNFA N as above, let $N' = (Q, \Sigma, \delta', Q'_0, F')$ be another XNFA, with transition matrix $M' = A^{-1}MA$ for some non-singular $n \times n$ matrix A, and let Q'_0 and F' be such that $v(Q'_0) = v(Q_0)A$ and $v(F')^T = A^{-1}v(F)^T$. Then

$$\begin{aligned}\Delta'(w_k) &= v(Q'_0)(M')^k v(F')^T \\ &= v(Q_0)A(A^{-1}MA)^k A^{-1}v(F)^T \\ &= v(Q_0)M^k v(F)^T \\ &= \Delta(w_k)\ .\end{aligned}$$

That is, for any existing XNFA with some transition matrix M, we can perform a change of basis to obtain $M' = A^{-1}MA$, $v(Q'_0) = v(Q_0)A$ and $v(F')^T = A^{-1}v(F)^T$. This represents the transition matrix, initial state vector and final state vector of a different XNFA that accepts the same language.

The following example illustrates how vector arithmetic in $GF(2)$ is used to determine if a word is accepted by the XNFA.

Example 2. *Let us consider again the XNFA N introduced in Example 1. The transition function δ is given by the following matrix:*

$$M = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix}$$

$Q_0 = \{q_0\}$ is encoded as $v(Q_0) = [\,1\,0\,0\,0\,]$, and let $F = \{q_0, q_1\}$ which is encoded as $v(F) = [\,1\,1\,0\,0\,]$. Suppose we have input word a^5.

Then the states reached after reading the entire input word is encoded by $v(Q_0)M^5$:

$$\begin{bmatrix} 1\,0\,0\,0 \end{bmatrix} \times \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix}^5 = \begin{bmatrix} 1\,1\,1\,0 \end{bmatrix}\ .$$

That is, $\delta(q_0, a^5) = \{q_0, q_1, q_2\}$. To determine $\Delta(a^5)$, we compute $v(Q_0)M^5F^T$:

$$\begin{bmatrix} 1\,1\,1\,0 \end{bmatrix} \times \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} = 0 \ .$$

We conclude that a^5 is not accepted by the XNFA N. Note that $F \subseteq \{q_0, q_1, q_2\}$, but $|F \cap \{q_0, q_1, q_2\}| = |\{q_0, q_1\}| = 2$, which confirms that the XDFA state reached on a^5 is not an accept state, since it does not contain an odd number of XNFA accept states. □

The following example shows how a change of basis affects the structures of XNFA and their equivalent XDFA. Since a change of basis does not affect the language accepted by the XNFA, we would expect the structures of the original XDFA and the new one to be similar, even if the structures of the XNFA are significantly different.

Example 3. *Let N be a 4-state unary XNFA with transition matrix M as given in Example 2, and let $Q_0 = \{q_0\}$ and $F = \{q_0, q_1\}$. The XNFA N is shown in Figure 3.1, while the equivalent XDFA N_D obtained via the subset construction is shown in Figure 3.2. As pointed out in Example 2, note that a^5 is not accepted by N_D. Now, let A be the non-singular matrix given in Figure 3.3, and let $M' = A^{-1}MA$, $v(Q_0') = v(Q_0)A$ and $v(F) = A^{-1}v(F)$, so that $Q_0' = \{q_2, q_3\}$ and $F' = \{q_0, q_3\}$, with M' given in Figure 3.4. The resulting XNFA N' and its equivalent XDFA N_D' are shown in Figures 3.5 and 3.6. Note that N_D and N_D' clearly accept the same language.* □

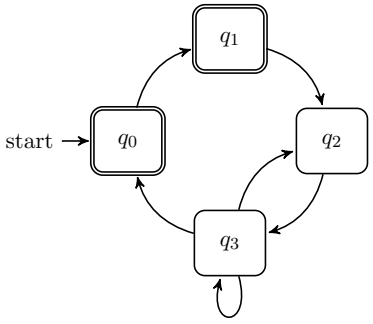


Figure 3.1: Example 3: N

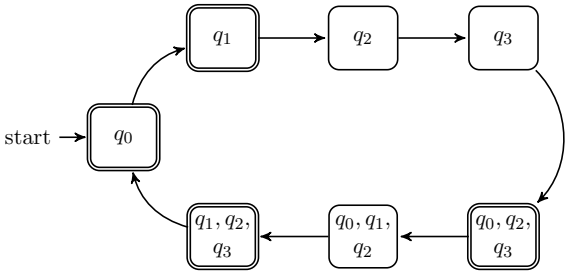


Figure 3.2: Example 3: N_D

$$A = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

Figure 3.3: Change matrix A

$$M' = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

Figure 3.4: Transition matrix M'

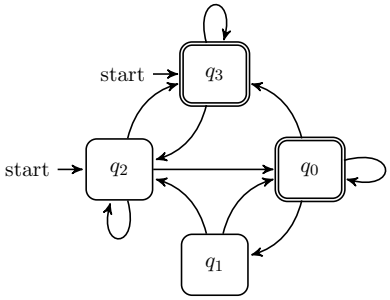


Figure 3.5: Example 3: N'

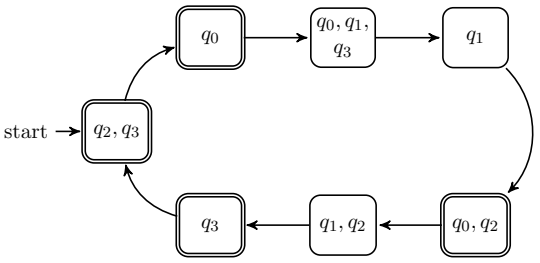


Figure 3.6: Example 3: N'_D

3.3 Unary XNFA: Polynomials and matrices in $GF(2)$

In this section, we continue to consider unary XNFA in order to present some useful properties, which we will be able to apply to XNFA with larger alphabets as well. It is possible to determine certain characteristics of unary XNFA by considering the properties of their transition matrices, especially with regards to the characteristic polynomial associated with each matrix. More specifically, unary XNFA have been shown to have equivalent cyclic behaviour to linear feedback shift registers (LFSRs) [32], and we now give some relevant results from [26] relating the cyclic properties of LFSRs, and hence of unary XNFA, to their associated matrices and characteristic polynomials over $GF(2)$.

Any $n \times n$ matrix M over $GF(2)$ has a characteristic polynomial $c(X) = \det(XI - M)$. On the other hand, every polynomial $c(X)$ over $GF(2)$ is the characteristic polynomial of some matrix M of the form shown in Fig. 3.7.

The matrix M is said to be the *normal form matrix* of $c(X)$, and to be in *canonical form*.

Each $c(X)$ is also associated with a so-called *companion matrix*, as stated in the next theorem.

Theorem 1. *[26] Every matrix M over $GF(2)$ with characteristic polynomial $c(X)$ is similar*[2] *to a matrix M' of the form shown in Figure 3.8, where each of the*

[2]Two $n \times n$ matrices A and A' are similar if there exists some non-singular $n \times n$ matrix B so that $A' = B^{-1}AB$.

$$M = \begin{bmatrix} 0 & 1 & 0 & \cdots & \cdots & 0 & 0 \\ 0 & 0 & 1 & & & 0 & 0 \\ 0 & 0 & 0 & \ddots & & 0 & 0 \\ \vdots & \vdots & \vdots & & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & & & 1 & 0 \\ 0 & 0 & 0 & \cdots & \cdots & 0 & 1 \\ c_0 & c_1 & c_2 & \cdots & \cdots & c_{n-2} & c_{n-1} \end{bmatrix}$$

Figure 3.7: Normal form matrix of $c(X)$

$$M' = \left[\begin{array}{c|c|c|c} A_1 & 0 & \ldots & 0 \\ \hline 0 & A_2 & \ldots & 0 \\ \hline \vdots & \vdots & \ddots & \vdots \\ \hline 0 & 0 & \ldots & A_m \end{array}\right]$$

Figure 3.8: Block diagonal matrix of normal form matrices

submatrices A_i is a normal form matrix of a polynomial that is irreducible over $GF(2)$ or a power of a polynomial that is irreducible over $GF(2)$, and the 0's are 0-submatrices of appropriate sizes. The matrix M' is said to be the companion matrix of $c(X)$.

Each block in the companion matrix M of some $c(X)$ represents a smaller automaton of which the characteristic polynomial is irreducible or is a power of an irreducible polynomial. If $c(X)$ has different factors, so that M has more than one block on the diagonal, the automaton can be thought of as a composite machine, where cycles from different blocks combine to form new cycles. This is made more clear in Theorem 4 on page 18.

When the subset construction is used on some XNFA with state set Q to obtain an equivalent XDFA, the resulting XDFA states are subsets of Q. Given a set of states $Q = \{q_0, q_1, ..., q_{n-1}\}$, in the following lemma from [26], it will be convenient to refer to an XDFA state, namely $d_s \subseteq Q$, as $s = \langle s_{n-1}, s_{n-2}, \ldots, s_1, s_0 \rangle$, where $s_i = 1$ if $q_i \in d_s$ and 0 otherwise.

Lemma 2. *[26] Let M_σ be a transition matrix representing transitions on σ for some XNFA N, with characteristic polynomial $c_\sigma(X)$, and let M_σ be in canonical form. We may regard 2^Q as representing the set of all possible states of the equivalent XDFA N_D. Then, let f be a bijection of the states from 2^Q onto polynomials of degree $n-1$, such that f maps the state $s = \langle s_{n-1}, s_{n-2}, ..., s_1, s_0 \rangle$ into the polynomial*

$f(s) = s_{n-1}X^{n-1} + s_{n-2}X^{n-2} + \cdots + s_1X + s_0$. Then f maps the state $s \cdot M_\sigma$ into the polynomial $Xf(s) \bmod c_\sigma(X)$.

Lemma 2 provides a mapping between polynomials over $GF(2)$ and the states of XDFA that are obtained via subset construction on unary XNFA with normal form transition matrices. The XDFA state arrived at after a transition from state s on σ corresponds to the polynomial which results from multiplying (multiplying) $f(s)$ by X in the polynomial algebra of $GF(2)[X]$ modulo $c(X)$. We illustrate this in the next example.

Example 4. *Consider again the transition matrix for the XNFA given in Example 2 and the corresponding XDFA transition table, given below.*

	δ_{XDFA}	a
$\rightarrow$	$[q_0]$	$[q_1]$
	$[q_1]$	$[q_2]$
	$[q_2]$	$[q_3]$
	$[q_3]$	$[q_0, q_2, q_3]$
$\leftarrow$	$[q_0, q_2, q_3]$	$[q_0, q_1, q_2]$
$\leftarrow$	$[q_0, q_1, q_2]$	$[q_1, q_2, q_3]$
	$[q_1, q_2, q_3]$	$[q_0]$

$$M = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix}$$

The characteristic polynomial of the matrix is $c(X) = X^4+X^3+X^2+1$. We see, for example, that $\delta_D(\{q_0, q_1, q_2\}, a) = \{q_1, q_2, q_3\}$, and it is clear that this corresponds to $X(X^2 + X + 1) = X^3 + X^2 + X$. In the case of $\delta_D(\{q_0, q_2, q_3\}, a) = \{q_0, q_1, q_2\}$, we have $X(X^3 + X^2 + 1) = X^4 + X^3 + X$. However,

$$(X^4 + X^3 + X) \bmod (X^4 + X^3 + X^2 + 1) = X^2 + X + 1 \ ,$$

which is consistent with Lemma 2. □

Note that the number of polynomials over $GF(2)$ modulo some $c(X)$ of degree n is exactly 2^n, including the zero polynomial. This corresponds to 2^n subsets of a set of states of size n, including the empty set. By Lemma 2, for a given normal form $n \times n$ matrix, a transition from any subset of states to its successor corresponds to multiplying its associated polynomial by X. If this is done for each subset of states, or equivalently, each possible XDFA state, a cyclic structure emerges. Furthermore, since XDFA structures are preserved under a change of basis, each $c(X)$ over $GF(2)$ is associated with a specific cycle structure, induced by any matrix for which it is the characteristic polynomial.

In this work we consider only matrices that are non-singular, since the cycle structures induced by singular matrices consist of cycles with various transient heads [26].

From a descriptional complexity point of view, such matrices are not interesting, because they do not result in large XDFA. In Example 7 on page 22 we illustrate why this is the case.

The following lemma shows that we therefore only need to consider polynomials that do not have X as a factor.

Lemma 3. *The normal form matrix of a polynomial over $GF(2)$ is singular if and only if X is a factor of the polynomial.*

Proof. Let $c(X)$ be some polynomial over $GF(2)$ of degree n. The coefficient of X^0 is zero if and only if X is a factor of $c(X)$, and so in the normal form matrix, $M_{0,n-1} = 0$. Then $\det(M) = 0$, which is equivalent to M being singular [5]. □

Since all similar matrices yield the same cycle structure, we can exclude all matrices of which the characteristic polynomial has X as a factor.

Theorem 4 below describes the possible cycle structures associated with polynomials that do not have X as a factor. Specifically, the properties of the characteristic polynomial of a unary XNFA N allow conclusions about the possible lengths and number of the cycles of states of its equivalent XDFA N_D (see [20] in particular, as well as for example [5; 26; 32]). The choice of initial states for an XNFA determines which cycle in its polynomial cycle structure represents the equivalent XDFA. We say that some $c(X)$ is the characteristic polynomial of some XNFA, if it is the characteristic polynomial of its transition matrix.

We give the following definition from [26], which is used in the theorem that follows.

Definition 1. *[26] For any monic*[3] *polynomial $f(X)$ over $GF(2)$, the* period *of $f(X)$ is defined to be the least integer k such that $f(X)$ divides $X^k - 1$.*

In the next theorem, the empty cycle is the cycle that contains the "empty" XDFA state, or the empty subset of XNFA states, which corresponds to the zero polynomial. Since $0 \cdot X = 0$, according to Lemma 2, we would have $\delta_D(\emptyset, a) = \emptyset$. However, the notion of an empty state need not be considered in the context of XDFA states obtained via the subset construction, and so we only consider the empty state in the context of the cycle structure of the polynomial, and not as part of a possible XDFA.

Theorem 4. *[26] Let $c(X)$ be a polynomial of degree n over $GF(2)$ that does not have X as a factor.*

[3] A monic polynomial is a polynomial of the form $X^n + c_{n-1}X^{n-1} \cdots c_1 X + c_0$; that is, a univariate polynomial for which the coefficient of the highest order term is 1.

- *If $c(X)$ is a primitive irreducible polynomial over $GF(2)$[4], then $c(X)$ has a single cycle of length $2^n - 1$, as well as the empty cycle of length 1.*

- *If $c(X)$ is an irreducible but not primitive polynomial over $GF(2)$, then $c(X)$ has $(2^n - 1)/b$ cycles of length b, where b is a factor of $2^n - 1$, as well as the empty cycle of length 1. In fact, b is the period of $c(X)$.*

- *If $c(X) = \phi(X)^m$ is the power of an irreducible polynomial $\phi(X)$ with degree d over $GF(2)$, we let k_i be the period of $\phi(X)^i$ for each i such that $1 \leq i \leq m$. Then $c(X)$ has the empty cycle of length 1, and all the non-empty states lie on μ_i distinct cycles of length k_i, where*

$$\mu_i = (2^{di} - 2^{d(i-1)})/k_i \ , \quad \text{for } 1 \leq i \leq m \ .$$

- *If $c(X)$ is a reducible polynomial over $GF(2)$, consider its companion matrix. For each cycle of length k_i induced by block A_i in the companion matrix and for each cycle of length k_j induced by block A_j, $c(X)$ has $\gcd(k_i, k_j)$ cycles of length $\operatorname{lcm}(k_i, k_j)$.*

We now give an example of each of these cases.

Example 5. *Consider the following four polynomials:*

$$\begin{aligned} c_a(X) &= X^4 + X + 1 \\ c_b(X) &= X^4 + X^3 + X^2 + X + 1 \\ c_c(X) &= X^4 + X^2 + 1 \\ c_d(X) &= X^4 + X^3 + X^2 + 1 \end{aligned}$$

By Theorem 4, we would expect the cycle structures as listed in Table 3.4.

The polynomials $c_a(X)$ and $c_b(X)$ are relatively straight forward as they are irreducible. The polynomial $c_a(X)$ is primitive and leads to a single cycle, while $c_b(X)$ is non-primitive, and hence leads to several cycles of the same length. Let us look more closely at $c_c(X)$ and $c_d(X)$, which are both reducible polynomials. The polynomial $c_c(X)$ is a power of the irreducible polynomial $\phi(X) = X^2 + X + 1$. First we set $i = 1$. The period of $(X^2+X+1)^1$ is the least integer k_1 such that $(X^2+X+1)^1$ divides $X^{k_1} - 1$. We have $X^2 + X + 1 \mid X^3 - 1$, and so $k_1 = 3$. Then we have

$$\begin{aligned} \mu_1 &= (2^{2\cdot 1} - 2^{2\cdot 0})/3 \\ &= (4 - 1)/3 = 1 \ . \end{aligned}$$

[4]A primitive polynomial of degree n over $GF(2)$ generates all elements in $GF(2^n)$. That is, a polynomial over $GF(2)$ is primitive if it has a root α such that $\{0, 1, \alpha, \alpha^2, ..., \alpha^{2^n-1}\}$ is the field $GF(2^n)$.

Table 3.4: Example 5: cycle structures of polynomials

Polynomial	Factors	Cycles
$c_a(X)$	$X^4 + X + 1$	1 cycle of length 15, empty cycle of length 1
$c_b(X)$	$X^4 + X^3 + X^2 + X + 1$	3 cycles of length 5, empty cycle of length 1
$c_c(X)$	$(X^2 + X + 1)^2$	1 cycle of length 3, 2 cycles of length 6, empty cycle of length 1
$c_d(X)$	$(X + 1)(X^3 + X + 1)$	1 cycle of length 1, 2 cycles of length 7, empty cycle of length 1

Hence, $c_c(X)$ has one cycle of length three. We now set $i = 2$, then $k_2 = 6$, since $(X^2 + X + 1)^2 \mid X^6 - 1$. Then we have

$$\begin{aligned}\mu_1 &= (2^{2\cdot 2} - 2^{2\cdot 1})/6 \\ &= (16 - 4)/6 = 2 \ .\end{aligned}$$

Hence, $c_c(X)$ has two cycles of length six.

The polynomials $c_d(X)$ is a reducible polynomial with factors $\phi_1(X) = X + 1$ and $\phi_2(X) = X^3 + X + 1$. Besides the empty cycles, which we designate ε_1 and ε_2, these two factors induce one cycle of length one and one cycle of length $2^3 - 1 = 7$, respectively. The cycles combine in the following way to produce new cycles:

1. *$X + 1$ and ε_2: $\gcd(1, 1)$ cycles of length $\operatorname{lcm}(1, 1)$, i.e. one cycle of length one*
2. *$X + 1$ and $X^3 + X + 1$: $\gcd(1, 7)$ cycles of length $\operatorname{lcm}(1, 7)$, i.e. one cycle of length seven*
3. *ε_1 and ε_2: $\gcd(1, 1)$ cycles of length $\operatorname{lcm}(1, 1)$, i.e. one cycle of length one*
4. *ε_1 and $X^3 + X + 1$: $\gcd(1, 7)$ cycles of length $\operatorname{lcm}(1, 7)$, i.e. one cycle of length seven*

Note that two cycles of length one are induced, of which one must be the empty cycle. □

The following example shows that if δ is extended to be a function from sets of states to sets of states, the resulting transition table corresponds to the cycle structure predicted by the characteristic polynomial of the transition matrix.

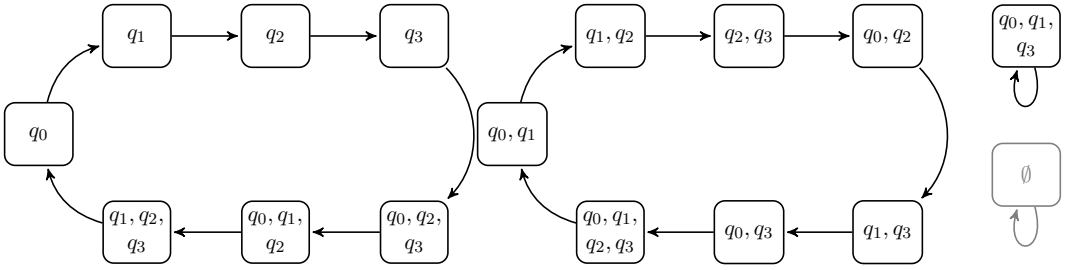


Figure 3.9: Example 6: cycle structure

Example 6. *In Examples 1 and 2, the transition function δ of the XNFA N was only defined for states that can be reached from the initial state q_0. If we consider the extension of δ as $\delta : 2^Q \times \Sigma \to 2^Q$, which is, in fact, equivalent to δ_D of the equivalent XDFA N_D, the possible transitions are as shown in Table 3.5. The resulting cycle structure is shown in Figure 3.9. Note that, since the transition matrix M of N has characteristic polynomial $c(X) = X^4 + X^3 + X^2 + 1$, the cycle structure is exactly as described in Example 5 for $c_d(X)$.*

Table 3.5: Example 6: expanded definition of δ_D

δ_D	a
$[q_0]$	$[q_1]$
$[q_1]$	$[q_2]$
$[q_2]$	$[q_3]$
$[q_3]$	$[q_0, q_2, q_3]$
$[q_0, q_1]$	$[q_1, q_2]$
$[q_0, q_2]$	$[q_1, q_3]$
$[q_0, q_3]$	$[q_0, q_1, q_2, q_3]$
$[q_1, q_2]$	$[q_2, q_3]$
$[q_1, q_3]$	$[q_0, q_3]$
$[q_2, q_3]$	$[q_0, q_2]$
$[q_0, q_1, q_2]$	$[q_1, q_2, q_3]$
$[q_0, q_1, q_3]$	$[q_0, q_1, q_3]$
$[q_0, q_2, q_3]$	$[q_0, q_1, q_2]$
$[q_1, q_2, q_3]$	$[q_0]$
$[q_0, q_1, q_2, q_3]$	$[q_0, q_1]$

Now, the characteristic polynomial of M is $c(X) = X^4 + X^3 + X^2 + 1 = (X + 1)(X^3 + X + 1)$. Given Theorem 4, this is exactly the cycle structure we would expect. The empty cycle, which necessarily has length 1, is always included in the calculation in Theorem 4, and is shown in grey in Figure 3.9. However, we need not consider it, except in the context of determining the complete cycle structures associated with

characteristic polynomials of XNFA. Furthermore, we can verify that each transition from a state that maps to a polynomial $f(X)$, as described in Lemma 2, is to the state that maps to $Xf(X)$. For example, $[q_2, q_3]$ maps to $X^3 + X^2$. Then

$$\begin{aligned} X(X^3 + X^2) \bmod (X^4 + X^3 + X^2 + 1) &= X^4 + X^3 \bmod (X^4 + X^3 + X^2 + 1) \\ &= X^2 + 1 \ . \end{aligned}$$

$X^2 + 1$ maps to $[q_0, q_2]$ and we see in Table 3.5 that $\delta([q_2, q_3]) = [q_0, q_2]$. For examples of the cycle structures of $c_a(X)$, $c_b(X)$ and $c_c(X)$ discussed in Example 5, see Appendix A, Example 31. □

The following example shows the divergent behaviour of polynomials that have X as a factor. We choose an arbitrary polynomial $c(X)$ that does not have X as a factor, and hence leads to a cycle structure where no transient heads are present; that is, where every state occurs in a cycle. Then, we illustrate how multiplication by X results in structures where the cycles remain the same size as with $c(X)$, while the possible transient heads increase in size and number. This illustrates why such structures are not interesting from a state complexity point of view.

Example 7. *Let $c(X) = X^3 + X^2 + X + 1$, and let $c' = Xc(X) = X^4 + X^3 + X^2 + X$ and $c'' = X^2c(X) = X^5 + X^4 + X^3 + X^2$, and let M, M' and M'' be their respective normal form matrices. The cycle structure of $c(X)$ is shown in Figure 3.10. The cycle structure of $c'(X)$, shown in Figure 3.11, is similar, except that for each state $d = [q_{i_1}, q_{i_2}, .., q_{i_k}]$ in the structure of $c(X)$, there is a state $d' = [q_{i_1+1}, q_{i_2+1}, .., q_{i_k+1}]$, and one other state leading to d'. The parts of the structure that also occur in the structure of $c(X)$ are shown in black, while the rest are shown in grey. Again, the cycle structure of $c''(X)$ is similar, with the common parts shown in black again. Now, each "extra" state, d', has two other states with transitions leading to it, shown in grey. The only way to reach the states shown in grey is by choosing one of them as the initial XDFA state, and then only one branch leading to the cycle reached, which means that the cycle structures of polynomials that have X as a factor do not lead to maximal XDFA sizes.* □

In this section we showed how the behaviour of a unary XNFA is characterised to a large extent by the characteristic polynomial of its transition matrix. In fact, changing the structure of the XNFA by adding or removing transitions and states changes the characteristic polynomial in ways that are not immediately evident. For example, an n-state XNFA with a primitive characteristic polynomial would have an equivalent XDFA that consists of a single cycle of length $2^n - 1$. Adding a single transition may result in an n-state XNFA with a reducible characteristic polynomial

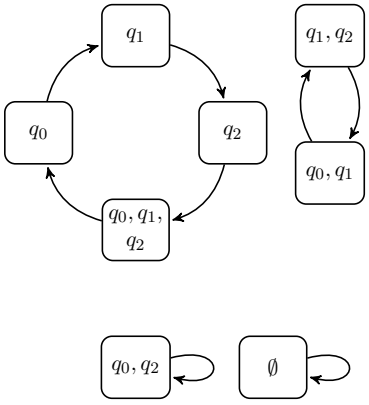


Figure 3.10: Example 7: cycle structure of $c(X)$

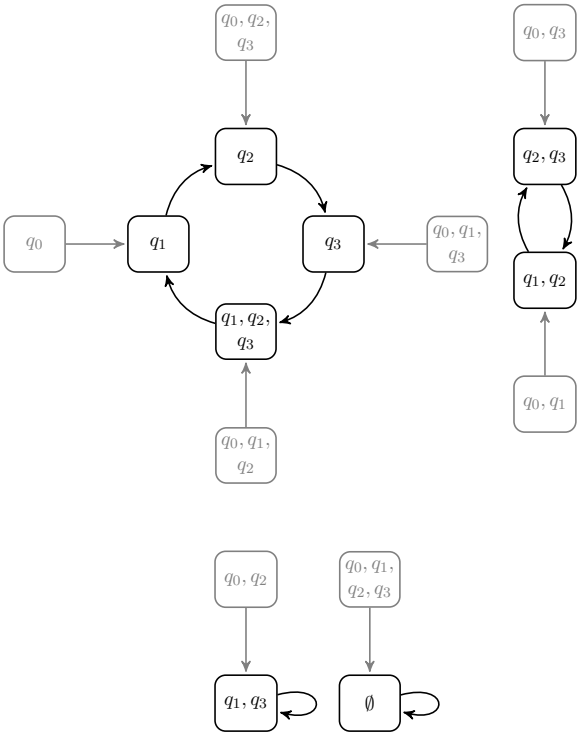


Figure 3.11: Example 7: cycle structure of $c'(X)$

of which the cycle structure consists of several cycles that are significantly shorter than $2^n - 1$. Hence, small changes to the structure of an XNFA may cause profound changes in its behaviour. Therefore, the approach followed here is to study the behaviour of XNFA primarily via their characteristic polynomials, instead of reasoning about their graph structures. For an example of such a profound change caused by adding a single transition, see Appendix A, Example 32.

3.4 Unary XNFA: linear recurrences over $GF(2)$

Unary XNFA also encode linear recurrences over $GF(2)$. In this section we show how this serves to give some information on how XDFA states occur together in cycles.

Since the structure of an XDFA is cyclic, for any state d_k of the XDFA that is reached after k letters have been read, there is some integer l so that, if $v(d_k) = v(Q_0)M^k$ for some k, then $v(d_k) = v(Q_0)M^{l+k}$. That is, l is the length of the cycle to which d_k belongs.

We introduce the notion of linear recurrences with respect to XNFA to provide more information about how XDFA states occur together in a cycle. A linear recurrence over a finite field has a characteristic polynomial [21]. Specifically, the

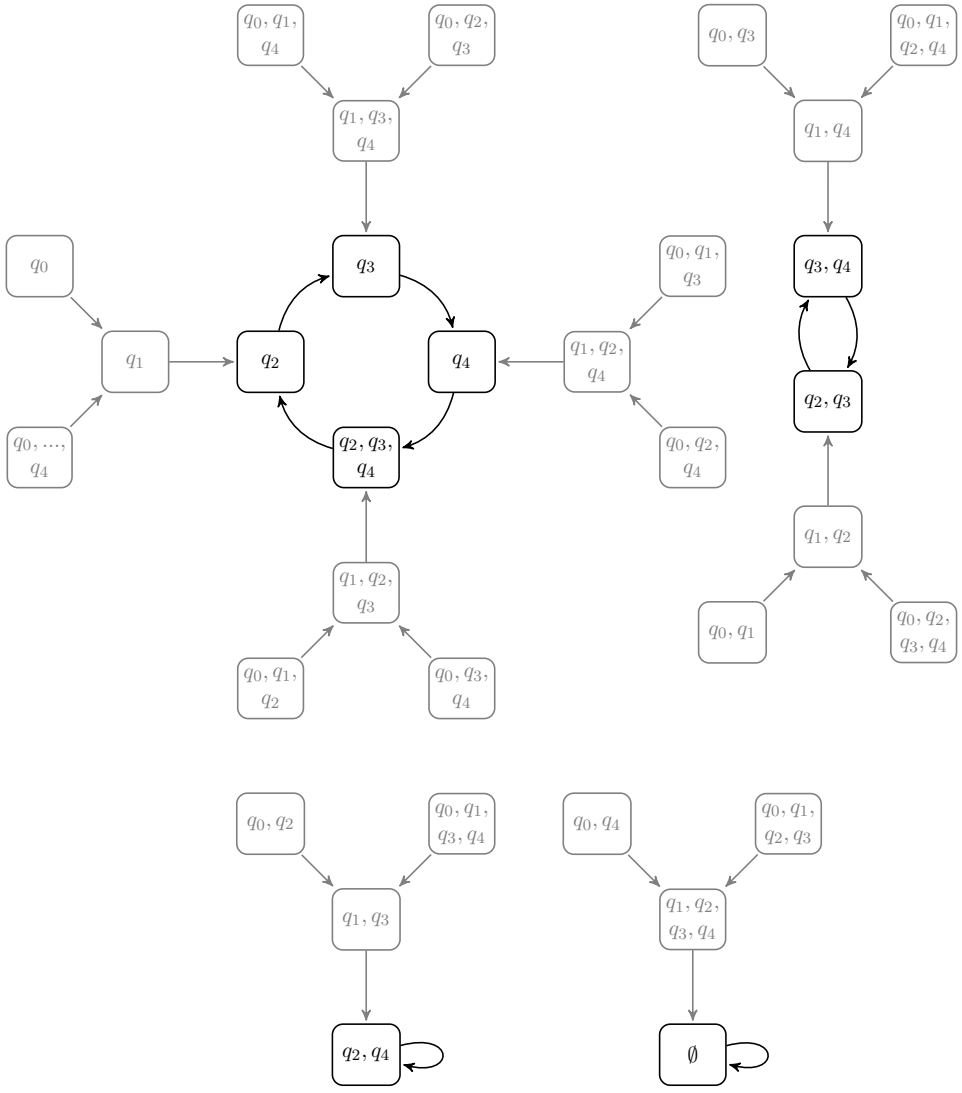


Figure 3.12: Example 7: cycle structure of $c''(X)$

polynomial $c(X) = X^n + c_{n-1}X^{n-1} + \cdots + c_0$ characterises the linear recurrence $s_t = c_{n-1}s_{t-1} + c_{n-2}s_{t-2} + \cdots + c_0 s_{t-n}$. Let $c(X)$ be the characteristic polynomial of

1. a transition matrix M for an n-state XNFA N, and also for
2. a linear recurrence over $GF(2)$, namely $s_t = \bigoplus_{i=1}^{n} c_{n-i}s_{t-i}$.

Let $\bar{s}_t = [s_{t_0}\ s_{t_1}\ \cdots\ s_{t_{n-1}}]$ be a vector of length n of elements in $GF(2)$. Then,

$$\begin{aligned}[s_{t_0}\ s_{t_1}\ \cdots\ s_{t_{n-1}}] = c_{n-1}[s_{t_0-1}\ s_{t_1-1}\ \cdots\ s_{t_{n-1}-1}]+\\ c_{n-2}[s_{t_0-2}\ s_{t_1-2}\ \cdots\ s_{t_{n-1}-2}]+\\ \vdots\\ +c_0[s_{t_0-n}\ s_{t_1-n}\ \cdots\ s_{t_{n-1}-n}]\ .\end{aligned} \tag{3.1}$$

That is, $\bar{s}_t = c_{n-1}\bar{s}_{t-1} + c_{n-2}\bar{s}_{t-2} + \cdots + c_0\bar{s}_{t-n}$.

Let $\bar{s}_0 = v(Q_0)$. The linear recurrence and the behaviour of the XNFA are both characterised by $c(X)$, so $\bar{s}_1 = v(Q_0)M$. In general $\bar{s}_k = v(Q_0)M^k$. We therefore have

$$\begin{aligned}v(d_t) &= v(Q_0)M^t\\ &= \bar{s}_t\\ &= c_{n-1}\bar{s}_{t-1} + c_{n-2}\bar{s}_{t-2} + \cdots + c_0\bar{s}_{t-n}\\ &= c_{n-1}v(Q_0)M^{t-1} + c_{n-2}v(Q_0)M^{t-2} + \cdots + c_0v(Q_0)M^{t-n}\\ &= c_{n-1}v(d_{t-1}) + c_{n-2}v(d_{t-2}) + \cdots + c_0v(d_{t-n})\ .\end{aligned} \tag{3.2}$$

Therefore, $d_t = \bigoplus_{i=1}^{n} c_{n-i}d_{t-i}$. That is, any state in a cycle is the symmetric difference sum of a certain number of states in the same cycle, and given the cycle, these states can be determined by inspecting the characteristic polynomial of the transition matrix.

3.4.1 Notation

In this work we let $\bar{s}_i$ refer to either the vector representing some set of states, or the set of states themselves, depending on the context. We use the symbol $\oplus$ and its sigma notation equivalent $\bigoplus$ to denote the boolean XOR operation when applied to boolean ones and zeroes, and the symmetric difference set operation when applied to sets, and specifically sets of states.

Example 8. *Let N be the XNFA in Example 6, with the cycle structure of its characteristic polynomial $c(X) = X^4 + X^3 + X^2 + 1$ shown in Figure 3.13. The corresponding linear recurrence is $s_t = s_{t-1} + s_{t-2} + s_{t-4}$. We choose an arbitrary state as $\bar{s}_t$, say $[q_2, q_3]$, and note that, equivalently,*

$$\begin{aligned}\bar{s}_t &= \bar{s}_{t-1} + \bar{s}_{t-2} + \bar{s}_{t-4}\\ [q_2, q_3] &= [q_1, q_2] \oplus [q_0, q_1] \oplus [q_0, q_3]\\ [\,0\,0\,1\,1\,] &= [\,0\,1\,1\,0\,] + [\,1\,1\,0\,0\,] + [\,1\,0\,0\,1\,]\ .\end{aligned}$$

□

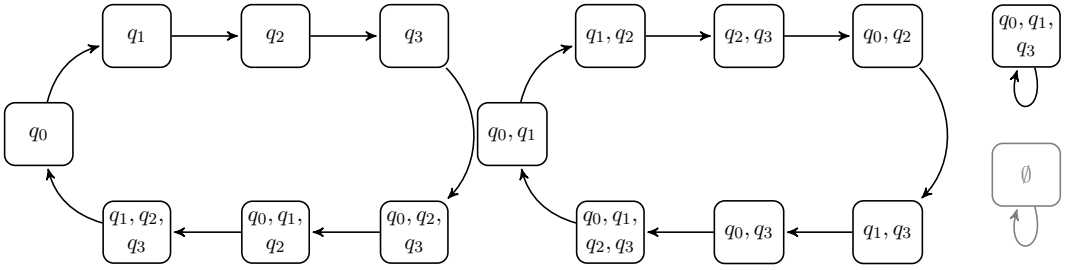


Figure 3.13: Example 8: cycle structure

3.5 Binary and r-ary XNFA

For unary XNFA, a single matrix encodes all transitions. In the case of binary and r-ary XNFA, each symbol is associated with a matrix, which encodes the transitions on the symbol in the same way as for the unary case. That is, if N is an XNFA for which $\Sigma = \{\sigma_1, \sigma_2, ..., \sigma_r\}$, then δ is encoded using r binary matrices, M_{σ_1}, M_{σ_2}, ..., M_{σ_r}, which encode transitions on σ_1, σ_2, ..., σ_r, respectively.

Example 9. *Let $N = (Q, \Sigma, \delta, Q_0, F)$ be a binary XNFA with $\Sigma = \{a, b\}$ and with δ as given in Table 3.6. Let $Q_0 = \{q_0\}$ and let $F = \{q_1, q_2\}$.*

Table 3.6: Example 9: transition function δ

δ	a	b
q_0	$\{q_1\}$	$\{q_1\}$
q_1	$\{q_2\}$	$\{q_2\}$
q_2	$\{q_0, q_1, q_2\}$	$\{q_0, q_2\}$

We encode transitions on a and b as matrices M_a and M_b, respectively, shown in Figures 3.14 and 3.15. These matrices lead to the cycle structures shown in Figures 3.16 and 3.17. Finally, the equivalent XDFA obtained via the subset construction is shown in Figure 3.18. Notice that the cycle structures associated with each matrix are present in the binary XDFA. □

$$M_a = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Figure 3.14: Example 9: transitions on a

$$M_b = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 1 \end{bmatrix}$$

Figure 3.15: Example 9: transitions on b

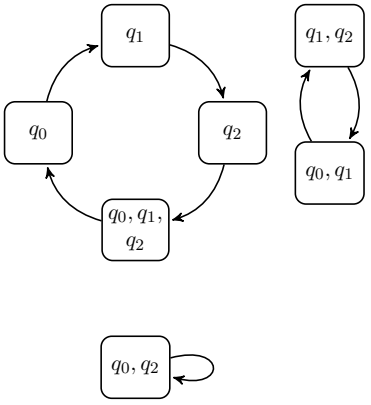


Figure 3.16: Example 9: transitions on a

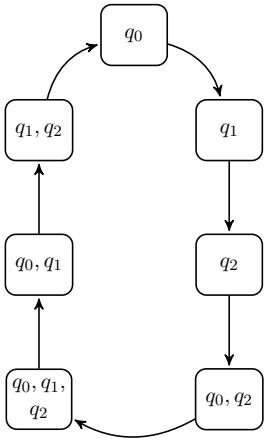


Figure 3.17: Example 9: transitions on b

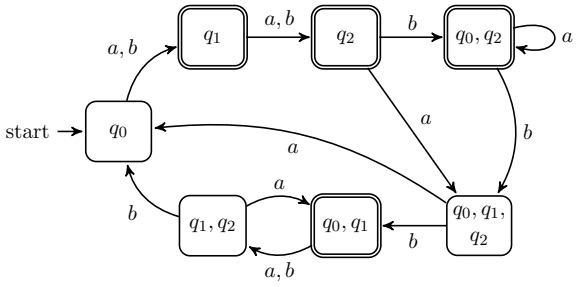


Figure 3.18: Example 9: binary XDFA

As with unary XNFA, it is possible to obtain equivalent r-ary XNFA by performing a change of basis on the transition matrices and initial and final state vectors. Let $N = (Q, \Sigma, \delta, Q_0, F)$ be an r-ary XNFA, with δ encoded by r matrices, M_{σ_1},M_{σ_2},...,M_{σ_r}, and let $N' = (Q, \Sigma, \delta', Q_0', F')$ be another r-ary XNFA, where $v(Q_0') = v(Q_0)A$, $v(F') = A^{-1}v(F)$, and $M'_{\sigma_i} = A^{-1}M_{\sigma_i}A$ for $1 \leq i \leq r$.

Let $w = \sigma_{i_1}\sigma_{i_2}\ldots\sigma_{i_k}$ be a word of length k, where σ_{i_j} represents the j-th letter of w. Then $\Delta(w) = v(Q_0)M_{\sigma_{i_1}}M_{\sigma_{i_2}}\ldots M_{\sigma_{i_k}}v(F)^T$, and

$$\begin{aligned}
\Delta'(w) &= v(Q_0')M'_{\sigma_{i_1}}M'_{\sigma_{i_2}}\ldots M'_{\sigma_{i_k}}v(F')^T \\
&= (v(Q_0)A)(A^{-1}M_{\sigma_{i_1}}A)(A^{-1}M_{\sigma_{i_2}}A)\ldots(A^{-1}M_{\sigma_{i_k}}A)(A^{-1}v(F)) \\
&= v(Q_0)M_{\sigma_{i_1}}M_{\sigma_{i_2}}\ldots M_{\sigma_{i_k}}v(F)^T \\
&= \Delta(w)\ .
\end{aligned}$$

In Example 9, the resulting XDFA contains all $2^n - 1$ non-empty states, primarily because the transitions on b form a cycle that reaches them all. Clearly, however, it is

conceivable that a binary XDFA may not reach all the non-empty states for different cycle structures associated with each symbol. We will present such examples in Chapter 4.

3.6 Generalised acceptance for XNFA

In the previous sections, and as is usual for finite state automata, XNFA were defined as having a single set of final states, F, that were interpreted as accepting states. Specifically, for XNFA, a word w is accepted if an odd number of paths labeled w end in final states. If zero or an even number of paths lead to final states, w is rejected.

The notion of multiple sets of final states has been explored for traditional NFA in the context of self-verifying NFA (SV-NFA) [1; 13; 16], as well as so-called Don't Care automata (dcNFA) [22]. In the case of self-verifying automata, there are two sets of final states, namely accept states and reject states, and it is required that for any word w, at least one path labelled w leads to a final state and no other path labelled w leads to a different kind of final state, so that every word is explicitly accepted or rejected by the automaton [16]. Don't Care automata also have accept states and reject states, but there is no requirement that every word have at least one path leading to final state, although no two paths with the same label can lead to different kinds of final states [22].

This raises the question of whether and how the notion of two sets of final states can be applied to XNFA, and in which ways this can be further generalised and applied to specific issues related to XNFA. We explore the notion of self-verifying XNFA in Chapter 4 and we define $\star$-XNFA in Chapter 5 and show how they can be used to succinctly represent a larger number of languages than can be succinctly represented by XNFA. For now we define *generalised acceptance XNFA*, of which SV-XNFA and $\star$-XNFA are instances.

Definition 2. *A generalised acceptance XNFA (GA-XNFA) is 5-tuple $N = (Q, \Sigma, \delta, Q_0, \mathcal{F})$, where Q represents the states of the automaton, Σ the finite alphabet over which it is defined, $\delta : Q \times \Sigma \to 2^Q$ the transitions between states, and Q_0 the set of initial states. Finally, $\mathcal{F} = \{F^1, F^2, \ldots, F^k\}$ is a finite set of sets of final states. Any input word w is accepted or not with respect to each $F \in \mathcal{F}$.*

Example 10. *Let N be a 3-state binary GA-XNFA with δ given in Table 3.6 on page 26. Let $\mathcal{F} = \{F^1, F^2\}$, where $F^1 = \{q_0, q_2\}$ and $F^2 = \{q_1, q_2\}$. The GA-XNFA N is shown in Figure 3.19, with states belonging to F^1 indicated by a double border and states belonging to F^2 indicated by a thick border. Note that q_2 belongs to both. The equivalent GA-XDFA N_D is given in Figure 3.20.*

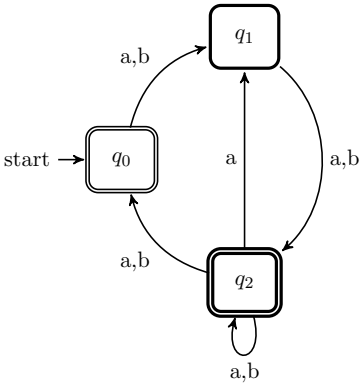


Figure 3.19: Example 10: N

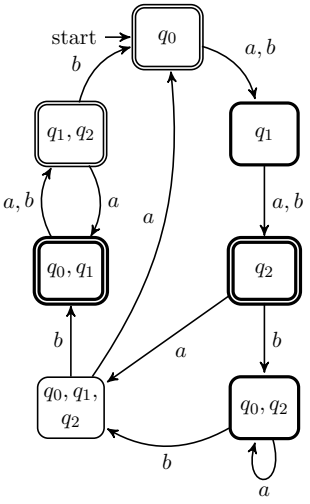


Figure 3.20: Example 10: N_D

We see, for example, that a^2 is accepted with respect to both F^1 and F^2, while a^3 is rejected with respect to both. Furthermore, a^2b is rejected with respect to F^1 and accepted with respect to F^2. □

Each set of final states in $\mathcal{F}$ therefore defines a language. We will see in the next chapter that self-verifying XNFA have two sets of final states, namely F^a and F^r, which define an accept language, say $\mathcal{L}^a$, and the reject language, say $\mathcal{L}^r$, respectively. The notion of self-verification implies that $\mathcal{L}^a \cap \mathcal{L}^r = \emptyset$. In Chapter 5 we explore some the possibilities of set operations on the languages associated with each set of final states.

Chapter 4

Self-verifying symmetric difference automata

In this chapter we consider self-verification as an instance of generalised acceptance. The notion of self-verification has been defined for typical (union) NFA [1; 13; 16], and here we consider whether and which ways it can be applied to XNFA.

Self-verifying nondeterministic finite automata (SV-NFA) have two sets of final states, namely accept states and reject states, while any non-final state is said to be neutral. That is, any path through the automaton leads to one of these three kinds of states, and it is required that for any word w presented to the SV-NFA, at least one path leads to a final state, and no two paths for w lead to different kinds of final states. In other words, every possible input is explicitly accepted or rejected by the automaton [16]. Rejection is not the result of a failure to reach an accept state as is usual for NFA; instead, it is required that a reject state be reached.

We can think of these requirements as a specific instance of generalised acceptance conditions, which we will apply to XNFA. The first question that arises is whether self-verification is possible for XNFA, and if so, under which circumstances. We consider the question of descriptional complexity for self-verifying XNFA (SV-XNFA) and see that it differs from XNFA. In this chapter we will define SV-XNFA in general, and first consider the properties of unary SV-XNFA in Section 4.1, before establishing a tight bound on the state complexity of unary SV-XNFA in Section 4.2. We turn to the state complexity of the binary and non-unary cases in Section 4.3, giving an upper bound of $2^n - 1$ and a lower bound of 2^{n-1}.

4.1 Properties of unary SV-XNFA

Having established relevant properties of XNFA in Chapter 3, we now turn to the question of self-verification for XNFA. Self-verification requires that any word be

either explicitly accepted or explicitly rejected, but not both. We call this the *SV-requirement*. The first question that arises is whether and when it is possible to choose a set of accept states and a set of reject states for an XNFA in such a way that the SV-requirement is met.

First, we give the following definition for SV-NFA.

Definition 3. *[1; 16] A self-verifying nondeterministic finite automaton (SV-NFA) is a 6-tuple $N = (Q, \Sigma, \delta, Q_0, F^a, F^r)$, where Q, Σ, δ and Q_0 are defined as for standard NFA. Here, $F^a \subseteq Q$ and $F^r \subseteq Q$, are the sets of accept and reject states, respectively. The remaining states, namely the states belonging to $Q \setminus (F^a \cup F^r)$, are called neutral states. For each input string w in Σ^*, it is required that there exist at least one path ending in either an accept or a reject state; that is, for each string w, we have $\delta(Q_0, w) \cap (F^a \cup F^r) \neq \emptyset$, and there is no string w such that both $\delta(Q_0, w) \cap F^a$ and $\delta(Q_0, w) \cap F^r$ are nonempty.*

Since any SV-NFA either accepts or rejects any string $w \in \Sigma^*$ explicitly, its equivalent DFA $N = (Q_D, \Sigma, \delta_D, Q_0, F^a_D, F^r_D)$ must do so too. The path for each w in a DFA is unique, so each state in the DFA is either an accept or reject state. Hence, for any DFA state d, there is some SV-NFA state $q_r \in d$ such that either $q_r \in F^a$, and consequently $d \in F^a_D$, or some $q_r \in F^r$, and consequently $d \in F^r_D$. Since each state in the DFA is a subset of states of the SV-NFA, accept and reject states cannot occur together in a DFA state. That is, if d is a DFA state, then for any $p, q \in d$, if $p \in F^a$ then $q \notin F^r$ and vice versa.

Based on this definition, we give the following definition for SV-XNFA.

Definition 4. *A self-verifying symmetric difference finite automaton (SV-XNFA) is a 6-tuple $N = (Q, \Sigma, \delta, Q_0, F^a, F^r)$, where Q, Σ, δ and Q_0 are defined as for XNFA, and F^a and F^r are defined as follows: F^a and F^r represent the accept states and reject states, respectively, and each state in the SV-XDFA equivalent to N must contain an odd number of states from either F^a or F^r, but not both.*

We refer to the equivalent DFA $N_D = (Q_D, \Sigma, \delta_D, Q_0, F^a_D, F^r_D)$ obtained via subset construction on an SV-XNFA as an SV-XDFA, in order to emphasise the presence of the SV-requirement and that this is determined via parity acceptance.

The choice of F^a and F^r for a given SV-XNFA N is called an *SV-assignment* of N. An SV-assignment where either F^a or F^r is empty, is called a *trivial SV-assignment*. Otherwise, if both F^a and F^r are nonempty, the SV-assignment is non-trivial. We say that a matrix M has an SV-assignment if some XNFA with M as its transition matrix has an SV-assignment.

Note that the SV-requirement for XNFA implies that if a state in the SV-XDFA of an SV-XNFA N contains an odd number of states from F^a, it may also contain

an even number of states from F^r, and so belongs to F^a_D, and vice versa. Parity is not applied to neutral states, so that any state in the SV-XDFA may contain any number of neutral states from N.

Implicit in the definition of SV-NFA is the requirement that $F^a \cap F^r = \emptyset$. This follows from the nature of NFA as being $\cup$-NFA, since it would simply violate the SV-requirement if some path in an SV-DFA led to a state that was both an accept state and a reject state. However, due to the parity nature of XNFA, this is not necessarily the case for SV-XNFA.

We now show that Definition 4 is essential to being able to perform a change of basis on the transition matrix of some unary SV-XNFA to obtain an equivalent SV-XNFA, as for XNFA.

Let $N = (Q, \Sigma, \delta, Q_0, F^a, F^r)$ be a unary SV-XNFA, with transition matrix M, and let $N' = (Q, \Sigma, \delta', Q'_0, F'^a, F'^r)$ be a unary SV-XNFA, with transition matrix $M' = A^{-1}MA$ for some non-singular $n \times n$ matrix A. Let Q'_0 and F' be such that $v(Q'_0) = v(Q_0)A$, $v(F'^a)^T = A^{-1}v(F^a)^T$ and $v(F'^r)^T = A^{-1}v(F^r)^T$. We let Δ^a refer to the *acceptance weight* of some word w_k. Then

$$\begin{aligned}\Delta'^a(w_k) &= v(Q'_0)(M'^k)v(F'^a)^T \\ &= v(Q_0)A(A^{-1}MA)^kA^{-1}v(F^a)^T \\ &= v(Q_0)(M^k)v(F^a)^T \\ &= \Delta^a(w_k)\ .\end{aligned}$$

Similarly, $\Delta'^r(w_k) = \Delta^r(w_k)$, where Δ^r refers to the *rejection weight* of w_k. We have therefore successfully performed a change of basis on N to produce N' which accepts the same language.

Just as for XNFA, this property is important for operations such as minimisation. However, if we chose F^a and F^r so that $F^a \cap F^r = \emptyset$, we have no guarantee that $F'^a \cap F'^r = \emptyset$. This means that for SV-XNFA, it makes sense to think of certain states as being both accept states and reject states. For example, if for some state d in the SV-XDFA N'_D obtained via the subset construction on N' there was some $q \in d$ such that $q \in F'^a \cap F'^r$, when counting the number of accept states in d as well as the number of reject states, q would contribute to both counts. Given that we know that N' accepts the same language as N, we know that counting in this way, the number of accept states would be odd and reject states would be even, or vice versa.

We call this kind of acceptance GF(2)-acceptance, to emphasise that it is defined in order to preserve the equivalence between XNFA and weighted automata over GF(2). We may state this more formally.

Lemma 5. *Given GF(2)-acceptance, for any n-state unary XNFA N with transition matrix M, if N has an SV-assignment, then there is an XNFA N' with transition matrix M' similar to M, where N and N' accept the same language. Consequently, if N has an SV-assignment, then so does N'.*

The following example illustrates such a change of basis for a unary SV-XNFA.

Example 11. *Let $N = (Q, \Sigma, \delta, Q_0, F^a, F^r)$ be a unary SV-XNFA with $Q_0 = \{q_0\}$, $F^a = \{q_0, q_1\}$ and $F^r = \{q_2, q_3\}$ and with its transition matrix being the normal form matrix M for the polynomial $c(X) = X^4 + X^3 + X^2 + 1$ given in Figure 4.1. Let the matrices A and M' (also shown in Figure 4.1) be related to M in the sense that $M' = A^{-1}MA$.*

Let $N' = (Q, \Sigma, \delta', Q'_0, F'^a, F'^r)$ be a unary XNFA with transition matrix M'. Let Q'_0, F'^a and F'^r be encoded by $v(Q'_0) = v(Q_0)A$, $v(F'^{aT}) = A^{-1}v(F^{aT})$ and $v(F'^{rT}) = A^{-1}v(F^{rT})$. That is, $Q'_0 = \{q_0, q_3\}$, $F'^a = \{q_1, q_3\}$ and $F'^r = \{q_0, q_3\}$.

The equivalent XDFA N_D and N'_D are shown in Figures 4.2 and 4.3 respectively, where double borders indicate accept states, and thick borders indicate reject states. Note that state $[q_0, q_3]$ in N'_D contains an odd number of accept states, namely q_3, and an even number of reject states, namely q_0 and q_3, and therefore it is an accept state. Clearly, N_D and N'_D accept the same language. □

$$M = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix} \quad A = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix} \quad M' = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Figure 4.1: Example 11: matrices M, A and M'

From the above discussion, it follows that for some SV-XDFA state $d \subseteq Q$, there must be a set of states $Q^F \subseteq Q$, such that for any $q^F \in Q^F$, if $q^F \in F^a$ then $q^F \notin F^r$ and vice versa. In other words, there must be some elements in d which "swing" the counts of accept and reject states in d so that one is odd and the other is even. The following lemma formalises the above discussion, which makes it possible, given any XDFA cycle, to determine whether an SV-assignment is possible.

Lemma 6. *Let $(d_1, d_2, ..., d_m)$ be a cycle representing a unary XDFA, obtained via the subset construction from some XNFA N, where $d_i \subseteq Q$ for $1 \leq i \leq m$, and Q is the set of states of N. The cycle has an SV-assignment if and only if for some*

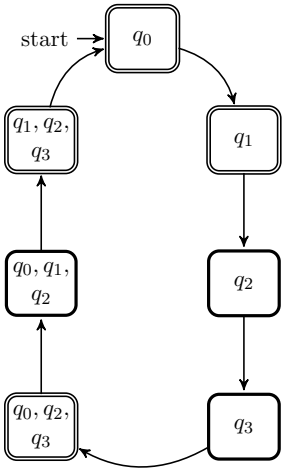


Figure 4.2: Example 11: N_D

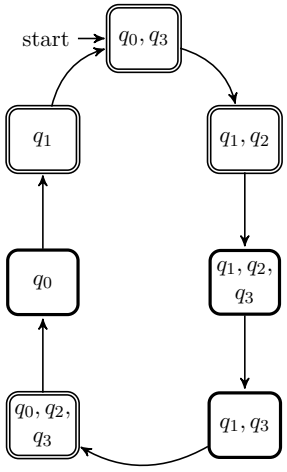


Figure 4.3: Example 11: N'_D

some choice of $Q^F \subseteq Q$*, where* $p_j = 1$ *for all* $q_j \in Q^F$ *and* $p_j = 0$ *otherwise, then the following holds:*

$$\bigwedge_{i=1}^{m} \bigoplus_{q_j \in d_i} p_j = 1 \ . \tag{4.1}$$

Proof. The expression in Equation 4.1 can only evaluate to 1 if every XDFA state d_i contains an odd number of XNFA states that result in a value of 1. This means that for some choice of Q^F, an odd number of its elements must be present in every XDFA state. □

Since $p \wedge p = p$, we also have the following corollary of Lemma 6.

Corollary 7. *For* $1 \leq k \leq m$*,*

$$\bigwedge_{i=1}^{m} \bigoplus_{q_j \in d_i} p_j = \bigwedge_{i=1}^{m} \bigoplus_{q_j \in d_i} p_j \ \wedge \bigoplus_{q_j \in d_k} p_j \ . \tag{4.2}$$

We assign some index $l > m$ *to a repeated state and generalise in the following way for any* $L \subseteq \{m+1, m+2, ...\}$*:*

$$\bigwedge_{i=1}^{m} \bigoplus_{q_j \in d_i} p_j = \bigwedge_{i=1}^{m} \bigoplus_{q_j \in d_i} p_j \ \wedge \bigwedge_{l \in L} \bigoplus_{q_j \in d_l} p_j \ . \tag{4.3}$$

Given GF(2)-acceptance and some XDFA cycle, Q^F represents those states of the XNFA that must belong to either F^a or F^r but not both for the cycle to have an SV-assignment. That is, every XDFA state must contain an odd number of states that contribute to the count of either F^a or F^r but not both, so that one but not both of the counts sum to an odd number. The states that contribute only to one count are the states that belong to Q^F.

However, given any F^a and F^r for which $Q^F \neq \emptyset$, a different SV-assignment may be found by letting $F'^a = F^a \cap Q^F$ and $F'^r \cap Q^F$, or equivalently, $F'^a = F^a \setminus F^r$ and $F'^r = F^r \setminus F^a$. That is, since any state in Q^F belongs only to one of the two sets of final states, $F'^a = F^a \cap Q^F$ consists only of those states in F^a that do not occur in F^r and vice versa. This would ensure that $F'^a \cap F'^r = \emptyset$, which is an SV-assignment, although the language accepted by the automaton would be different.

Consider again N' in Example 11, which has $F'^a = \{q_1, q_3\}$ and $F'^r = \{q_0, q_3\}$. Since q_0 and q_1 do not occur in both sets, $Q^F = \{q_0, q_1\}$. Let N'' be the SV-XNFA that is identical to N' except that $F''^a = F'^a \cap Q^F = \{q_1\}$ and $F''^r = F'^r \cap Q^F = \{q_0\}$. Figure 4.4 gives the equivalent SV-XDFA of N''. The cycle is identical to the SV-XDFA in Figure 4.3, but the final states are different because of a different choice of accept and reject states. Note, however, that the presence of q_0 and q_1, the so-called "swing states", in every state of the cycle guarantees that F''^a and F''^r constitute an SV-assignment.

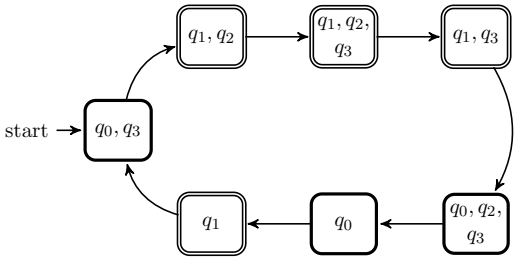


Figure 4.4: Example 11: N''_D

In other words, for any SV-assignment given GF(2)-acceptance, another SV-assignment may be found where $F'^a \cap F'^r = \emptyset$. We call the requirement that $F'^a \cap F'^r = \emptyset$, disjunctive acceptance. Note that any SV-assignment on disjunctive acceptance is by definition also an SV-assignment on GF(2)-acceptance. Therefore, for the purposes of establishing descriptional complexity bounds for unary SV-XNFA, we need consider only GF(2)-acceptance going forward.

We illustrate Lemma 6 with an example.

Example 12. *Consider the XDFA cycle shown in Figure 4.5. This cycle leads to the following expression:*

$$(p_0 \oplus p_3) \wedge (p_1 \oplus p_2) \wedge (p_1 \oplus p_2 \oplus p_3) \wedge (p_1 \oplus p_3) \wedge (p_0 \oplus p_2 \oplus p_3) \wedge p_0 \wedge p_1. \quad (4.4)$$

If we choose $Q^F = \{q_0, q_1\}$, the expression becomes the following:

$$(1 \oplus 0) \wedge (1 \oplus 0) \wedge (1 \oplus 0 \oplus 0) \wedge (1 \oplus 0) \wedge (1 \oplus 0 \oplus 0) \wedge 1 \wedge 1 = 1. \quad (4.5)$$

We can choose $F^a = \{q_1\}$ *and* $F^r = \{q_0\}$*, while* $F^a = \{q_1, q_3\}$ *and* $F^r = \{q_0, q_3\}$ *is also an SV-assignment. These two SV-assignments correspond to the cycles as shown in Figures 4.4 and 4.3, respectively.* □

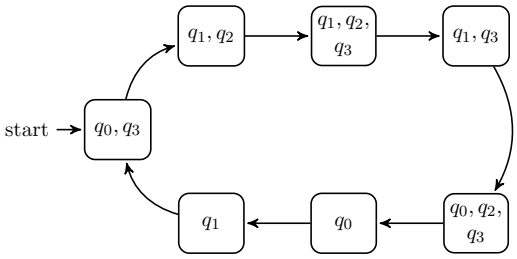


Figure 4.5: Example 12: XDFA cycle

In order to determine which properties of XDFA cycles (obtained via the subset construction from some unary XNFA N) allow for SV-assignments, we must be able to determine to some degree how the states of N are distributed within the cycles in its cycle structure. The following two lemmas shed more light on this.

Lemma 8. *The RHS (right hand side) of the linear recurrence* $s_t = c_{n-1}s_{t-1} + c_{n-2}s_{t-2} + \cdots + c_0 s_{t-n}$ *of a polynomial* $c(X) = X^n + c_{n-1}X^{n-1} + \cdots + c_1 X + c_0$ *has an odd number of terms if* $X+1$ *is a factor of* $c(X)$*, and an even number otherwise.*

Proof. Recall that in $GF(2)$, $a - b = a + b$. Hence, if $X + 1$ is a factor of $c(X)$, then 1 is a root, i.e. $(1)^n + c_{n-1}(1)^{n-1} + \cdots + c_1(1) + c_0 = 0$, hence $c(X)$ must have an even number of non-zero coefficients. Therefore, an odd number of coefficients among $c_{n-1}, c_{n-2}, ..., c_0$ must be non-zero. Similarly, if $X+1$ is not a factor, an even number of coefficients among $c_{n-1}, c_{n-2}, ..., c_0$ must be non-zero. □

Lemma 9. *Let* d_1 *be any state in an XDFA cycle of an equivalent XNFA with state set* Q *and let the cycle be characterised by some linear recurrence. That is, each state in the cycle is the symmetric difference sum of certain states in the cycle, as determined by the linear recurrence. For any* u, $d_{(u+1) \bmod m}$ *refers to the state following* d_u*, while* $d_{(u-1) \bmod m}$ *refers to the state preceding* d_u*.*

Let $\sigma_u = \bigoplus_{q_j \in d_u} p_j$ *for some choice of* $Q^F \subseteq Q$ *so that* $p_j = 1$ *if* $q_j \in Q^F$ *and* $p_j = 0$ *otherwise. Furthermore, let* $T \subseteq \{2, ..., n\}$ *be the set of indices such that* $d_1 = \bigoplus_{k \in T} d_k$*. Then*

$$\sigma_1 = \bigoplus_{k \in T} \sigma_k. \tag{4.6}$$

In the case where the cycle length $m \leq n$*, it is possible that* $d_{(1-i) \bmod m} = d_{(1-j) \bmod m}$ *for some* i, j*. We assign to the* l*-th duplicate of a state* d_k *(including any occurrences*

of d_1 itself) the index $lm + k$, referring to it as d_{lm+k}. This allows us to regard repeated XDFA states as distinct elements of T in order to reason accurately about the number of occurrences of any given XNFA state q_i and the corresponding value p_i in the equations that follow.

Proof. Let $r = |T|$ and consider the following:

$$\bigoplus_{k\in T} \sigma_k = \sigma_{k_1} \oplus \sigma_{k_2} \oplus \cdots \oplus \sigma_{k_r} \tag{4.7}$$

$$= \bigoplus_{q_j \in d_{k_1}} p_j \oplus \bigoplus_{q_j \in d_{k_2}} p_j \oplus \cdots \oplus \bigoplus_{q_j \in d_{k_r}} p_j \ . \tag{4.8}$$

Since $d_1 = \bigoplus_{k\in T} d_k$, we know that each p_j on the RHS of the above equation occurs an odd number of times if $q_j \in d_1$, and an even number of times otherwise. Therefore,

$$\bigoplus_{k\in T} \sigma_k = \bigoplus_{q_j \in d_1} p_j = \sigma_1 \ .$$

□

We are now in a position to present the following theorem which gives a necessary condition for the possibility of SV-assignments for unary XNFA.

Theorem 10. *A unary n-state XNFA N with characteristic polynomial $c(X) = X^n + c_{n-1}X^{n-1} + \cdots + c_1X + c_0$ has no SV-assignment, if $X + 1$ is not a factor of $c(X)$.*

Proof. From the discussion in Section 3.4, we know that the state transition behaviour of N is described by $s_t = c_{n-1}s_{t-1} + c_{n-2}s_{t-2} + \cdots + c_0 s_{t-n}$.

That is, in its equivalent XDFA N_D of length m, each state is the $\oplus$-sum of some number of states in its cycle. Consider any cycle of N_D and let d_1 be any state in the cycle. Let $\mathbb{T} = \{2, ..., n\}$ and let $T_1 \subseteq \mathbb{T}$ be the set of indices so that

$$d_1 = \bigoplus d_i \text{ where } i \in T_1 \ .$$

If $m > n$, we use Equation 4.1 from Lemma 6 on page 33 as well as Lemma 9 on page 36 to determine if the cycle has an SV-assignment.

$$\begin{aligned}
\bigwedge_{i=1}^{m} \bigoplus_{q_j \in d_i} p_j &= \bigwedge_{i=1}^{m} \sigma_i \\
&= \sigma_1 \wedge \bigwedge_{i\in T_1} \sigma_i \ \wedge \bigwedge_{i\in \mathbb{T}\setminus T_1} \sigma_i \\
&= \bigoplus_{i\in T_1} \sigma_i \ \wedge \bigwedge_{i\in T_1} \sigma_i \ \wedge \bigwedge_{i\in \mathbb{T}\setminus T_1} \sigma_i \ .
\end{aligned}$$

If $m \leq n$, we let $K = \{i \in T_1 |\, i > m\}$ and use Equation 4.3 from Corollary 7 and Lemma 9 similarly:

$$\begin{aligned}\bigwedge_{i=1}^{m} \bigoplus_{q_j \in d_i} p_j &= \bigwedge_{i=1}^{m} \bigoplus_{q_j \in d_i} p_j \ \wedge \bigwedge_{i \in K} \bigoplus_{q_j \in d_i} p_j \\ &= \bigwedge_{i=1}^{m} \sigma_i \wedge \bigwedge_{i \in K} \sigma_i \\ &= \sigma_1 \wedge \bigwedge_{i \in T_1} \sigma_i \ \wedge \bigwedge_{i \in \mathbb{T} \setminus T_1} \sigma_i \\ &= \bigoplus_{i \in T_1} \sigma_i \ \wedge \bigwedge_{i \in T_1} \sigma_i \ \wedge \bigwedge_{i \in \mathbb{T} \setminus T_1} \sigma_i \ .\end{aligned}$$

In both cases, if $c(X)$ does not have $X + 1$ as a factor, then by Lemma 8, $|T_1|$ is even. Therefore, $\bigoplus_{i \in T_1} \sigma_i \wedge \bigwedge_{i \in T_1} \sigma_i = 0$, and so the cycle does not have an SV-assignment. □

We illustrate Theorem 10 with the following example.

Example 13. *Let $c(X) = X^3 + X + 1$ be the characteristic polynomial of an XNFA N, of which the transition matrix M is given in Figure 4.6 and for which $Q_0 = \{q_0\}$.*

$$M = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Figure 4.6: Example 13: transition matrix

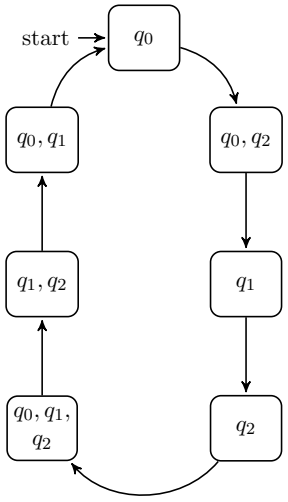


Figure 4.7: Example 13: XDFA cycle

Figure 4.7 shows the cycle of the equivalent XDFA. The linear recurrence characterised by $c(X)$ is $s_t = s_{t-2} + s_{t-3}$, and consequently we see that, for example, $[q_0] = [q_1, q_2] \oplus [q_0, q_1, q_2]$. Let $d_1 = [q_0]$, so that $d_5 = [q_0, q_1, q_2]$ and $d_6 = [q_1, q_2]$.

Hence, $\sigma_1 = p_0$*,* $\sigma_5 = (p_0 \oplus p_1 \oplus p_2)$ *and* $\sigma_6 = (p_1 \oplus p_2)$ *for some choice of binary values for* p_0*,* p_1 *and* p_2*, and so by Lemma 9,*

$$p_0 = (p_1 \oplus p_2) \oplus (p_0 \oplus p_1 \oplus p_2) \ .$$

To determine if the cycle has an SV-assignment, we evaluate the following expression from Lemma 6:

$$\begin{aligned} & p_0 \wedge (p_1 \oplus p_2) \wedge (p_0 \oplus p_1 \oplus p_2) \\ = \ & [(p_1 \oplus p_2) \oplus (p_0 \oplus p_1 \oplus p_2)] \wedge (p_1 \oplus p_2) \wedge (p_0 \oplus p_1 \oplus p_2) \\ = \ & (\sigma_6 \oplus \sigma_5) \wedge \sigma_6 \wedge \sigma_5 \ . \end{aligned}$$

For the entire expression to evaluate to 1*, the first term requires that only one of* σ_6 *or* σ_5 *have a value of* 1*, while the second and third terms require that both have a value of* 1*. Clearly, this is not possible, and so the expression evaluates to* 0 *for any choice of values for* p_0*,* p_1 *and* p_2*. Hence, by Lemma 6, no SV-assignment is possible.* □

Having given a necessary condition for non-trivial SV-assignments for XNFA, we now consider whether any sufficient conditions can be established.

To do this, we consider the cycle structure of the normal form matrix. We start with the following lemma, which proves a useful property of the cycles produced by any normal form matrix, namely that such cycles contain either odd sized states or even sized states, but not both.

Lemma 11. *Let* M *be the normal form matrix of some* $c(X) = (X+1)\phi(X)$*. Let* $\delta : 2^Q \times \Sigma \to 2^Q$ *be the transition function encoded by* M*. Since* δ *describes unary transitions, we let* $\Sigma = \{a\}$*. Then for any* $d \in 2^Q$*,* $|\delta_D(d, a)|$ *is odd if and only if* $|d|$ *is odd.*

Proof. M is an $n \times n$ matrix that has the form given below:

$$M = \begin{bmatrix} 0 & 1 & 0 & \cdots & \cdots & 0 & 0 \\ 0 & 0 & 1 & & & 0 & 0 \\ 0 & 0 & 0 & \ddots & & 0 & 0 \\ \vdots & \vdots & \vdots & & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & & & 1 & 0 \\ 0 & 0 & 0 & \cdots & \cdots & 0 & 1 \\ c_0 & c_1 & c_2 & \cdots & \cdots & c_{n-2} & c_{n-1} \end{bmatrix} .$$

We note again that in $GF(2)$, $a - b = a + b$ and hence, as $X + 1$ is a factor of $c(X)$, then 1 is a root, i.e. $(1)^n + c_{n-1}(1)^{n-1} + \cdots + c_1(1) + c_0 = 0$. Consequently, an

odd number among $c_0, c_1, \ldots, c_{n-2}, c_{n-1}$ are 1's. Therefore, we see that $|\delta(q_{n-1}, a)|$ is odd by inspecting the bottom row of M, while $|\delta(q_i, a)| = 1$ for $0 \leq i \leq n-2$. We consider two cases regarding d.

Case 1: $q_{n-1} \notin d$. Let $d = [q_{i_1}, q_{i_2}, \ldots, q_{i_k}]$. From the transition matrix it is clear that $\delta_D(d, a) = [q_{i_1+1}, q_{i_2+1}, \ldots, q_{i_k+1}]$, and hence $|d| = |\delta_D(d, a)| = k$. Hence, d transitions to a state of the same size.

Case 2: $q_{n-1} \in d$. Let $d = [q_{i_1}, q_{i_2}, \ldots, q_{i_k}, q_{n-1}]$. Suppose that $|d|$ is odd, and hence that k is even. Let $D = [q_{i_1}, q_{i_2}, \ldots, q_{i_k}]$. From the transition matrix we see that $\delta_D(d, a) = \delta_D(D, a) \oplus \delta_D([q_{n-1}], a)$.

We have $\delta(D, a) = [q_{i_1+1}, q_{i_2+1}, \ldots, q_{i_k+1}]$, and so $|\delta_D(D, a)| = |D| = k$, which is even. Also, $|\delta_D([q_{n-1}], a)|$ is odd. Therefore, $|\delta_D(D, a)| + |\delta_D([q_{n-1}], a)|$ is odd. The symmetric difference of two sets is the elements that occur in exactly one of the two sets. If some element occurs in both, then it is not in the symmetric difference. We can therefore "cancel out" elements occurring in both $\delta_D(D, a)$ and $\delta_D([q_{n-1}], a)$ in pairs, until only those remain that occur in exactly one of the sets. But successively removing pairs of elements from an odd number of elements leaves an odd number of elements. So $\delta_D(D, a) \oplus \delta_D([q_{n-1}], a)$ must be odd, and hence $|\delta_D(d, a)|$ is odd.

Similarly, if $|d|$ is even, then $|\delta_D(d, a)|$ is even.

□

We also present the following lemma, which states certain useful properties associated with the cycle containing the XDFA state $[q_0]$ given the normal form matrix of some $c(X) = (X+1)\phi(X)$.

Lemma 12. *Let $c(X) = (X+1)\phi(X)$ be a polynomial of degree n with non-singular normal form matrix M, and let N be a unary XNFA with transition matrix M and $Q_0 = \{q_0\}$. Then the equivalent XDFA N_D has the following properties:*

1. $|Q_D| \geq n$
2. *$|d|$ is odd for each $d \in Q_D$*
3. $[q_0], [q_1], ..., [q_{n-1}] \in Q_D$

Proof. Since $\delta(q_i, a) = \{q_{i+1}\}$ for all $i < n-1$, N_D contains the states $[q_0], [q_1], ..., [q_{n-1}]$, and therefore forms a cycle with at least n states. These states all have odd size, so by Lemma 11, all other states in the cycle must have odd size as well. □

The following theorem gives a sufficient condition for non-trivial SV-assignments for XNFA.

Theorem 13. *Any matrix M with characteristic polynomial $c(X) = (X+1)\phi(X)$ has an SV-assignment.*

Proof. Recall that a matrix is said to have an SV-assignment if any XNFA for which it is the transition matrix has an SV-assignment. Let M be a transition matrix with characteristic polynomial $c(X)$. Then M is similar to M_N [26], the normal form matrix of $c(X)$. That is, there exists some $n \times n$ matrix A so that $M = A^{-1}M_N A$. Consider the cycle in the cycle structure of M_N that contains the state $[q_0]$. By Lemma 12, the cycle contains all XDFA states consisting of a single XNFA state, and all states in the cycle each contain an odd number of XNFA states. Hence, if N_N is an XNFA with $Q_0 = \{q_0\}$, any choice of F^a and F^r so that $F^a \cap F^r = \emptyset$ and $F^a \cup F^r = Q$ will be a SV-assignment. Since M_N and M are similar, by Lemma 5, if N is an XNFA with transition matrix M, given the appropriate choice of initial states, N also has an SV-assignment. □

Finally, the following theorem follows directly from Theorems 10 and 13.

Theorem 14. *Any matrix M has an SV-assignment, if and only if its characteristic polynomial has $X+1$ as a factor.*

4.2 Descriptional complexity of unary SV-XNFA

Having established relevant properties of unary SV-XNFA, we now turn to the question of their state complexity. In the previous section, we established that, for the unary case, an SV-assignment is only and always possible for matrices of which the characteristic polynomials have $X+1$ as a factor. Therefore, we examine the properties of unary XNFA with such polynomials, and this leads us to conclude that $2^{n-1}-1$ is a tight bound on the state complexity of unary SV-XNFA.

Lemma 15. *For a unary XNFA with a characteristic polynomial $c(X)$ with degree n that has $X+1$ as a factor, the longest possible cycle has length $2^{n-1}-1$.*

Proof. Suppose $c(X)$ has two irreducible factors, $\phi_1(X) = X+1$ and $\phi_2(X)$, where $\phi_2(X)$ is an irreducible polynomial with degree $n-1$. By Theorem 4 on page 18, if $\phi_2(X)$ is primitive it has a single cycle of length $2^{n-1}-1$, and together with $X+1$ induces a cycle for $c(X)$ of the same length. If $\phi_2(X)$ is non-primitive, its cycle structure has cycles of length b where b is a factor of $2^{n-1}-1$, inducing cycles of length b for $c(X)$ together with $X+1$. Hence, the maximum cycle length is $2^{n-1}-1$.

Now suppose that $c(X)$ has three irreducible factors, $\phi_1 = X+1$, $\phi_2(X)$ of degree $k \leq n-2$ and $\phi_3(X)$ of degree $n-k-1$, with $k > n-k-1$. Cycles of $c(X)$ induced together with $X+1$ can only produce at most cycles of length $2^k - 1 < 2^{n-1} - 1$. Consider the cycle induced by $\phi_2(X)$ and $\phi_3(X)$, which, by Theorem 4 on page 18 the will have length $\text{lcm}(2^k-1, 2^{n-k-1}-1)$. Since the cycle will have greatest possible length if $2^k - 1$ and $2^{n-k-1} - 1$ are relatively prime, we assume this to be the case. The cycle induced has length $\text{lcm}(2^k - 1, 2^{n-k-1} - 1) = (2^k - 1)(2^{n-k-1} - 1)$. That is,

$$\begin{aligned}(2^k - 1)(2^{n-k-1} - 1) &= 2^{n-1} - 2^k - 2^{n-k-1} - 1 \\ &< 2^{n-1} - 1 \ .\end{aligned}$$

Cycles of $c(X)$ are induced by pairs of factors of $c(X)$, and so if $c(X)$ had more irreducible factors, they would have smaller degree and so would induce even shorter cycles. Therefore, $2^{n-1} - 1$ is the longest possible cycle for a polynomial $c(X)$ of degree n that has $X + 1$ as a factor. □

Lemma 12 on page 40 gives certain relevant properties of the cycle containing the state $[q_0]$ (as well as all other XDFA states of size one) given the normal form matrix of some $c(X) = (X + 1)\phi(X)$. The following lemma provides another useful property of this cycle.

Lemma 16. *Let $c(X) = (X+1)\phi(X)$ be a polynomial of degree n with non-singular normal form matrix M, and let $N = (Q, \Sigma, \delta, Q_0, F)$ be a unary XNFA with transition matrix M and $Q_0 = \{q_0\}$. Let $N_D = (Q_D, \Sigma, \delta_D, Q_{0,D}, F_D)$ be the equivalent XDFA obtained via the subset construction. For any $d \in Q_D$, let q_m be the XNFA state in d with the highest subscript m. Then $q_0 \in \delta_D(d, a^{n-m})$ and $q_0 \notin \delta_D(d, a^k)$ for any $0 < k < n - m$.*

Proof. From the transition matrix, $\delta(q_i, a) = \{q_{i+1}\}$ for $0 \leq i < n - 1$, and $q_0 \in \delta(q_{n-1}, a)$. Note that for any q_i, if $0 \leq i < n - 1$, then $q_0 \notin \delta(q_i, a)$.

Consequently, $q_0 \in \delta_D(d, a)$ if an only if $q_{n-1} \in d$. Hence, if $q_m = q_{n-1}$ for some d, then $a^{n-m} = a^{n-(n-1)} = a$ and $q_0 \in \delta_D(d, a)$, and therefore $q_0 \in \delta_D(d, a^{n-m})$.

Note also that, for any $d \in Q_D$, if $q_i \in d$ for $0 \leq i < n - 1$ is the state with the highest subscript in d, then q_{i+1} is the state in $\delta_D(d, a)$ with the highest subscript.

Therefore, if $q_m \neq q_{n-1}$ for some d, then $q_{n-1} \in \delta_D(d, a^{(n-1)-m})$, and so $q_0 \in \delta_D(d, a^{(n-m)})$. Also, $q_{n-1} \notin \delta_D(d, a^k)$ for any $0 < k < (n - 1) - m$, and so $q_0 \notin \delta_D(d, a^{k'})$ for any $0 < k' < n - m$. □

More informally, Lemma 16 provides a way of knowing the minimal non-zero number of transitions one must follow in the N_D described in order to reach a state containing q_0 from any state d.

We now establish that the cycle considered in Lemma 12 reaches the maximum cycle length for $c(X) = (X+1)\phi(X)$.

Lemma 17. *Let $c(X) = (X+1)\phi(X)$ be a polynomial of degree n with non-singular normal form matrix M and where $\phi(X)$ is a primitive polynomial. Let N be a unary XNFA with transition matrix M and $Q_0 = \{q_0\}$. Then N_D forms a cycle of length $2^{n-1} - 1$.*

Proof. We calculate the number and lengths of all cycles for $c(X)$. By Theorem 4, factors $X+1$ and $\phi(X)$ each induce a single cycle of length $2^m - 1$ with $m = 1$ and $m = n-1$ respectively, as well as a single cycle each of length 1, which is the so-called empty cycle ε. Therefore $c(X)$ has the following cycles:

- ε_{X+1} and $\varepsilon_{\phi(X)}$: $\gcd(1,1)$ cycle(s) of length $\mathrm{lcm}(1,1)$
- $X+1$ and $\varepsilon_{\phi(X)}$: $\gcd(1,1)$ cycle(s) of length $\mathrm{lcm}(1,1)$
- ε_{X+1} and $\phi(X)$: $\gcd(1, 2^{n-1}-1)$ cycle(s) of length $\mathrm{lcm}(1, 2^{n-1}-1)$
- $X+1$ and $\phi(X)$: $\gcd(1, 2^{n-1}-1)$ cycle(s) of length $\mathrm{lcm}(1, 2^{n-1}-1)$

Therefore, $c(X)$ has two cycles of length one, one of which is $\varepsilon_{c(X)}$, and two cycles of length $2^{n-1} - 1$. By Lemma 12, N_D must be a cycle of at least length n, so it must have length $2^{n-1} - 1$. □

We now show that the cycle in Lemma 12 always has an SV-assignment.

Theorem 18. *Let $c(X) = (X+1)\phi(X)$ be a polynomial of degree n with non-singular normal form matrix M. Then if N is the unary XNFA with transition matrix M and $Q = \{q_0\}$, any choice of F^a and F^r so that $F^a \cup F^r = Q$ and $F^a \cap F^r = \emptyset$ with F^a and F^r non-empty is an SV-assignment.*

Proof. From Lemma 12 it follows that the XNFA N of which the transition matrix is the companion matrix of $c(X)$ has a cycle of at least length n in which each state has odd size. Furthermore, $[q_0], [q_1], ..., [q_{n-1}]$ are all states in Q_D, so $q_0, q_1, ..., q_{n-1}$ must all be in either F^a or F^r.

Therefore, any choice of F^a and F^r so that $F^a \cup F^r = Q$ and $F^a \cap F^r = \emptyset$ with F^a and F^r non-empty will guarantee that each state in the XDFA contains an odd number of states from either F^a or F^r and zero or an even number of states from the other, and hence will be an SV-assignment. □

The following example is of a language that is accepted by a 4-state XNFA, but which requires $2^{4-1}-1$ XDFA states.

Example 14. *Let $\phi(X) = X^3+X+1$, which is a primitive polynomial. Now, let N be a unary XNFA with transition matrix M_a, where M_a is the normal form matrix of $c_a(X) = (X+1)\phi(X) = X^4+X^3+X^2+1$. Let $Q_0 = \{q_0\}$ and let $F^a = \{q_0\}$ and $F^r = \{q_1, q_2, q_3\}$. The matrix M_a is shown in Figure 4.8, while N and its equivalent XDFA N_D are shown in Figures 4.9 and 4.10. We have $\mathcal{L} = a^{7i+j}$ for $i \geq 0$ and $j \in \{0, 4, 5\}$, and as Figure 4.10 shows, exactly $2^{n-1}-1 = 2^{4-1}-1 = 7$ states are necessary to represent $\mathcal{L}$ with an XDFA.*

$$M_a = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix}$$

Figure 4.8: Example 14: transition matrix for a

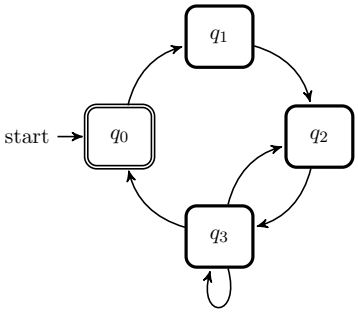


Figure 4.9: Example 14: N

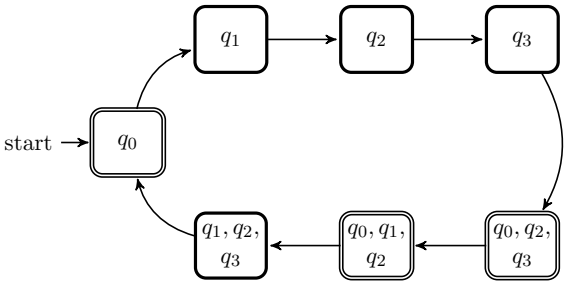


Figure 4.10: Example 14: N_D

□

Notice that due to the choice of $F^a = \{q_0\}$ and $F^r = Q \setminus \{q_0\}$, the XDFA cycle starts in $[q_0]$, which is an accept state, and only reaches the next accept state, namely $[q_0, q_2, q_3]$, after 4 transitions. After that, accept states occur more frequently in the cycle. We can see why this is by considering Lemma 16, which states that the number of transitions required to reach the next state containing q_0 depends on the highest subscript m of the current state: the higher the subscript m, the fewer transitions until a state containing q_0 is reached. This distance is maximal

when $m = 0$ in $[q_0]$. The proof of the following theorem relies on the fact that this pattern, i.e. two accept states separated by n transitions, occurs only once in the XDFA cycle, and hence the language accepted by the XDFA cannot be represented by a smaller XDFA cycle.

Theorem 19. *For any $n \geq 2$, there is a language $\mathcal{L}_n$ so that some n-state unary SV-XNFA accepts $\mathcal{L}_n$ and the minimal SV-XDFA that accepts $\mathcal{L}_n$ has $2^{n-1}-1$ states.*

Proof. Let $c(X) = (X+1)\phi(X)$ be a polynomial of degree n, where $\phi(X)$ is a primitive polynomial, and let M be its non-singular normal form matrix. Let N be an XNFA with transition matrix M and let $Q_0 = \{q_0\}$. By Theorem 13, N has an SV-assignment, and in fact, by Theorem 18, $F^a = \{q_0\}$ and $F^r = Q \setminus F^a$ is such an assignment. By Lemma 17, the N_D obtained via subset construction has $2^{n-1} - 1$ states.

We let $F^a = \{q_0\}$ and $F^r = Q \setminus F^a$. Then $\mathcal{L}_n = a^{(2^{n-1}-1)i+j}$ for $i \geq 0$ and $j \in J$, where J is some set of integers. Now, from the transition matrix of N it follows that $0, n \in J$, while $1, 2, ..., n-1 \notin J$, since $q_0 \in \delta(q_0, a^n)$ and $q_0 \notin \delta(q_0, a^m)$ for $m < n$. This means that, starting in q_0, another set of states containing q_0 is only reached after reading a^n.

If there is an N'_D with fewer than $2^{n-1} - 1$ states that accepts $\mathcal{L}_n$, then there must exist some $d_j \neq [q_0] \in Q_D$ such that $q_0 \in d_j$, $q_0 \in \delta_D(d_j, a^n)$ and there is no $m < n$ so that $q_0 \in \delta_D(d_j, a^m)$.

However, by Lemma 16, $q_0 \in \delta_D(d_j, a^{n-k})$, where k is the largest subscript of any state in d_j. Since $d_j \neq [q_0]$, it follows that $k > 0$, and so there is some $m = n-k < n$ such that $q_0 \in \delta_D(d_j, a^m)$. This is a contradiction, so no such d_j exists. Therefore, there is no N'_D with fewer than $2^{n-1} - 1$ states that accepts $\mathcal{L}_n$. □

We have shown that the maximum cycle length $2^{n-1} - 1$ for a unary SV-XNFA can be reached. We therefore conclude that $2^{n-1} - 1$ is a tight bound for the state complexity of unary SV-XNFA.

We have seen that $X + 1$ as a factor is a necessary and sufficient condition for polynomial to have an SV-assignment; that is, for a cycle in its cycle structure to have an SV-assignment. The following example sheds some further light on why this is the case. In the preceding proofs, we have made use of cycles associated with the normal form matrix of $c(X)$. Here, however, we consider the entire cycle structure of $c(X) = X^4+X^3+X^2+1$ with reference to the cycles associated with its block diagonal companion matrix, as described in Section 3.3 on page 16, in order to illustrate more clearly the effect of $X+1$ as a factor on the entire cycle structure.

Example 15. *Consider again the polynomial $c(X) = X^4+X^3+X^2+1$, which was given in Example 14 as an illustration of the tight bound for unary SV-XNFA. We*

can write $c(X)$ as the products of its factors, namely $c(X) = (X+1)(X^3+X+1)$. Figures 4.11 and 4.12 show the cycle structures associated with the normal form matrices of the two factors X^3+X+1 and $X+1$ respectively. For clarity, we name the state in the latter cycle q_3 instead of q_0.

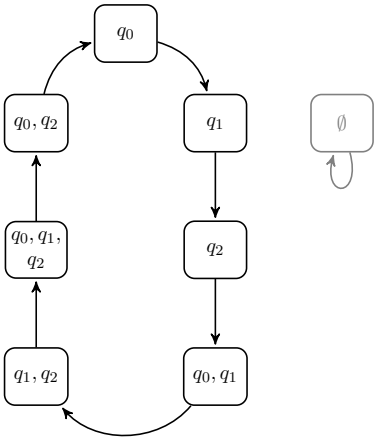


Figure 4.11: Example 15: cycles of X^3+X+1

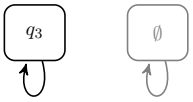


Figure 4.12: Example 15: cycles of $X+1$

Recall that in Section 3.3 on page 16, we noted that XNFA with reducible polynomials can be thought of as composite machines, where the cycles of the various factors combine into new cycles. Each block in the block diagonal companion matrix of a polynomial $c(X)$ represents the normal form matrix of a factor or a power of a factor of $c(X)$, contributing its own cycles. Stone [26] describes the cycles of the composite machine in terms of "running" pairs of cycles from different factors concurrently.

Let $\mathcal{C}_{a_1}$ and $\mathcal{C}_{a_2}$ refer to the cycles of length 7 and length 1 from the structure of X^3+X+1 respectively, and let $\mathcal{C}_{b_1}$ and $\mathcal{C}_{b_2}$ refer to the two cycles length 1, one containing the non-empty state and one containing the empty state, respectively, from the structure of $X+1$. We consider the cycle structure associated with the block diagonal matrix of $c(X) = (X+1)(X^3+X+1)$ with reference to these cycles:

1. *Cycles $\mathcal{C}_{a_2}$ and $\mathcal{C}_{b_2}$ are the two empty cycles, which when run concurrently results in another empty cycle.*

2. *When $\mathcal{C}_{a_1}$ is run concurrently with the empty cycle $\mathcal{C}_{b_2}$, the result is a copy of $\mathcal{C}_{a_1}$.*

3. *When $\mathcal{C}_{a_1}$ is run concurrently with $\mathcal{C}_{b_1}$, the result is almost a copy of $\mathcal{C}_{a_1}$: each state consists of the states contributed by $\mathcal{C}_{a_1}$ as well as the single state contributed by $\mathcal{C}_{b_1}$.*

4. *When the empty cycle $\mathcal{C}_{a_2}$ is run concurrently with $\mathcal{C}_{b_2}$, the result is a copy of $\mathcal{C}_{b_2}$.*

Figure 4.13 gives the resulting cycle structure of $c(X)$, which is associated with the block diagonal matrix of $c(X)$. We see that the leftmost cycle of length 7, which is a copy of the cycle $\mathcal{C}_{a_1}$ from $X^3 + X + 1$, cannot have an SV-assignment. On the one hand, it contains the states $[q_0]$, $[q_1]$ and $[q_2]$, and hence q_0, q_1 and q_2 must all belong to either F^a or F^r but not both. On the other hand, it contains $[q_0, q_1]$, which then either contains two (an even number of) states from F^a (or F^r), or one state each from F^a and F^r. In either case, the SV-condition is violated. However, if we consider the rightmost cycle of length 7, then any choice of F^a and F^r so that q_0, q_1 and q_2 belong to either F^a or F^r and where q_3 belongs to both F^a and F^r is an SV-assignment, for example $F^a = \{q_0, q_1, q_3\}$ and $F^r = \{q_2, q_3\}$. Such a choice guarantees that each XDFA state has an odd number of XNFA states from F^a and an even number of XNFA states from F^r, or vice versa.

Furthermore, the cycle which is a copy of $\mathcal{C}_{b_1}$ has a trivial SV-assignment, while the empty cycle trivially has none. □

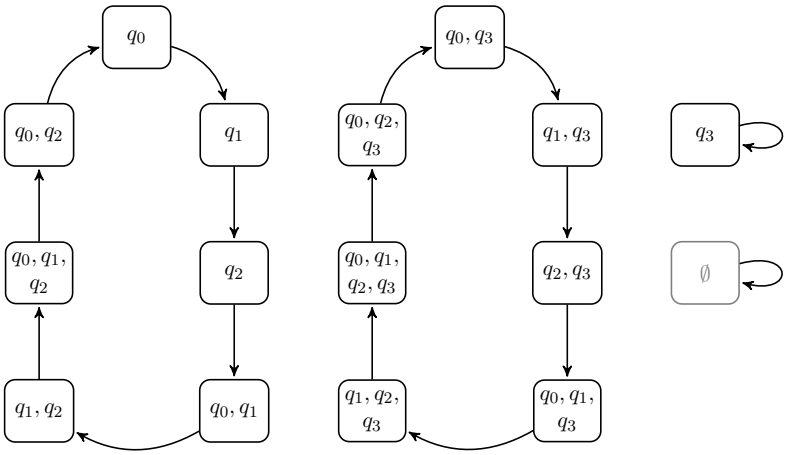


Figure 4.13: Example 15: structure of block diagonal matrix

One should not conclude from the above example that the presence of $X+1$ as a factor of some $c(X)$ always produces such neat copies of the cycle structures of the other factors of $c(X)$. Specifically, due to the cycle structures of polynomials that are powers of irreducible polynomials (see Theorem 4 on page 18), while the presence of $X+1$ guarantees the existence of cycles with SV-assignments, the resulting cycle structure of $c(X)$ may vary depending on the multiplicity of $X+1$ as a factor of $c(X)$.

4.3 Descriptional complexity of non-unary SV-XNFA

We now turn to non-unary SV-XNFA in order to establish state complexity bounds. Since binary and r-ary XNFA are in some sense the result of combining unary XNFA, we first establish some additional properties of unary XNFA, before showing that 2^{n-1} is a lower bound on state complexity for binary and r-ary SV-XNFA. The upper bound is simply $2^n - 1$, which is the size of the power set over a set of size n excluding the empty set, but this bound is known not to be tight, since an SV-assignment for an XDFA that reaches all $2^n - 1$ states is not possible [20].

Recall that according to Lemma 11 on page 39, cycles produced by a normal form matrix contain either odd sized states or even sized states, but not both. We now show that given r polynomials that all have $X + 1$ as a factor and their normal form matrices, any r-ary SV-XNFA with such transition matrices can have at most 2^{n-1} states.

Theorem 20. *Let M_{σ_1}, M_{σ_2}, ..., M_{σ_r} be the normal form matrices of r polynomials $c_{\sigma_1}(X) = (X+1)\phi_{\sigma_1}(X)$, $c_{\sigma_2}(X) = (X+1)\phi_{\sigma_2}(X)$, ..., $c_{\sigma_r}(X) = (X+1)\phi_{\sigma_r}(X)$, respectively, and let M_{σ_1}, M_{σ_2}, ..., M_{σ_r} be the transition matrices of some r-ary XNFA N with $\Sigma = \{\sigma_1, \sigma_2, ..., \sigma_r\}$ and $Q_0 = \{q_0\}$. Then the number of states in the equivalent XDFA N_D does not exceed 2^{n-1}. Furthermore, any choice of F^a and F^r such that $F^a \cup F^r = Q$ and $F^a \cap F^r = \emptyset$ is an SV-assignment.*

Proof. By Lemma 12, $|d|$ is odd for $d \in Q_D$ in the unary case, and by Lemma 11, for any symbol with a transition matrix of which the polynomial has $X + 1$ as a factor, a transition from an odd-sized XDFA state is to another odd-sized XDFA state. Since $Q_0 = \{q_0\}$ and $|\{q_0\}|$ is odd, and $c_{\sigma_1}(X)$, $c_{\sigma_2}(X)$,...,$c_{\sigma_r}(X)$ all have $X + 1$ as a factor, only odd-sized states are reachable on any transition. The number of XDFA states d such that $|d|$ is odd is $2^n/2 = 2^{n-1}$, and so N_D can have at most 2^{n-1} states. Since every XDFA state contains an odd number of XNFA states, any choice of F^a and F^r such that $F^a \cup F^r = Q$ and $F^a \cap F^r = \emptyset$ is an SV-assignment. □

The following lemma provides further information on the cycle structure induced by polynomials with $X + 1$ as a factor.

Lemma 21. *Let $c_\sigma(X) = (X+1)\phi(X)$. Then, in the normal form matrix M_σ of $c_\sigma(X)$, which is the transition matrix on some symbol σ for an XNFA, the state d_ϕ mapped to $\phi(X)$, as described in Lemma 2 on page 16, is contained in a cycle of length one, when considering only transitions on σ.*

Proof. Recall that $a + b = a - b$ in $GF(2)$ and consider the following:

$$\begin{aligned}(X+1)\phi(X) &= c_\sigma(X)\\ X\phi(X) + \phi(X) &= c_\sigma(X)\\ X\phi(X) &= \phi(X) + c_\sigma(X)\ .\end{aligned}$$

Therefore, $X\phi(X) = \phi(X)$ in the representation of $GF(2^n)$ as polynomials over GF(2) modulo $c_\sigma(X)$. By Lemma 2, this corresponds to $\delta_D(d_\phi, \sigma) = d_\phi$. □

Next, we show that for any $n \geq 4$, a binary SV-XNFA exists which consists of exactly 2^{n-1} states.

Lemma 22. *Let $\phi(X) = X^{n-1} + \phi_{n-2}X^{n-2} + \cdots + \phi_1 X + \phi_0$ be any primitive polynomial of degree $n-1$, where $n \geq 4$. Let N be a binary XNFA, and let the transition matrix on the alphabet symbol a be the normal form matrix of $c_a(X) = (X+1)\phi(X)$ and the transition matrix on the alphabet symbol b be the normal form matrix of $c_b(X) = X^n + \phi(X)$. Then the equivalent XDFA of the XNFA with $Q_0 = \{q_0\}$ contains exactly 2^{n-1} odd-sized states.*

Proof. Note that, unless $n \geq 4$, $c_a(X)$ and $c_b(X)$ (which have degree $n-1$) cannot be distinct polynomials, since there is only one primitive polynomial of degree 2, namely $X^2 + X + 1$.

We write $c_a(X)$ and $c_b(X)$ in the following way:

$$\begin{aligned}c_a(X) &= X^n + c_{n-1}X^{n-1} + \cdots + c_1 X + c_0\\ c_b(X) &= X^n + \phi_{n-1}X^{n-1} + \phi_{n-2}X^{n-2} + \cdots + \phi_1 X + \phi_0\ .\end{aligned}$$

Since $\phi(X)$ is primitive, it has no roots in GF(2), including 1, so it must have an odd number of non-zero terms. Therefore, by Lemma 2, $|d_\phi|$ is odd. Furthermore, $c_b(X)$ has an even number of non-zero terms, and so has 1 as a root. Consequently, $c_b(X)$ has $X + 1$ as a factor.

The transition matrices M_a and M_b are given in Fig. 4.14 and 4.15. Note that they are both non-singular. Let $Q_0 = \{q_0\}$. Then by Lemma 17, the cycle structure on a is equivalent to an XDFA cycle with $2^{n-1} - 1$ states, all of which, by Lemma 12, have odd size. Also, by Lemma 21, d_ϕ is not contained in this cycle. This means that on a, every odd-sized state in the XDFA is reached except for d_ϕ.

Now, from M_b it follows directly that $\delta_D([q_{n-1}], b) = d_\phi$. Furthermore, since $X + 1$ is a factor of c_b, every transition from an odd-sized state on b is to an odd-sized state. Consequently, the binary XNFA N is equivalent to an XDFA that reaches all 2^{n-1} odd-sized states and none other. □

$$M_a = \begin{bmatrix} 0 & 1 & 0 & \cdots & \cdots & 0 & 0 \\ 0 & 0 & 1 & & & 0 & 0 \\ 0 & 0 & 0 & \ddots & & 0 & 0 \\ \vdots & \vdots & \vdots & & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & & & 1 & 0 \\ 0 & 0 & 0 & \cdots & \cdots & 0 & 1 \\ c_0 & c_1 & c_2 & \cdots & \cdots & c_{n-2} & c_{n-1} \end{bmatrix}$$

Figure 4.14: Lemma 22: transition matrix for a

$$M_b = \begin{bmatrix} 0 & 1 & 0 & \cdots & \cdots & 0 & 0 \\ 0 & 0 & 1 & & & 0 & 0 \\ 0 & 0 & 0 & \ddots & & 0 & 0 \\ \vdots & \vdots & \vdots & & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & & & 1 & 0 \\ 0 & 0 & 0 & \cdots & \cdots & 0 & 1 \\ \phi_0 & \phi_1 & \phi_2 & \cdots & \cdots & \phi_{n-2} & \phi_{n-1} \end{bmatrix}$$

Figure 4.15: Lemma 22: transition matrix for b

Theorem 23. *For any $n \geq 4$, there is a language $\mathcal{L}_n$ so that some n-state binary SV-XNFA accepts $\mathcal{L}_n$ and the minimal SV-XDFA that accepts $\mathcal{L}_n$ has 2^{n-1} states.*

Proof. Let $c_a(X) = (X+1)\phi(X)$ and $c_b = X^n + \phi(X)$, where $\phi(X)$ is a primitive polynomial and let $c_a(X)$ and $c_b(X)$ have degree n. We construct an SV-XNFA N with n states of which the equivalent XDFA N_D has 2^{n-1} states as in Lemma 22, and let $F^a = \{q_0\}$ and $F^r = Q \setminus F^a$. Recall that for N, we have $\delta : Q \times \Sigma \to 2^Q$, and for N_D, we have $\delta_D : 2^Q \times \Sigma \to 2^Q$.

Let $\mathcal{L}_n^1 = a^{(2^{n-1}-1)i+j}$ for $i \geq 0$ and $j \in J$, where J is some set of integers, represent a subset of the language accepted by N that consists only of strings containing a. In the same way as on page 45, we prove that there is no N'_D with fewer than $2^{n-1}-1$ states that accepts $\mathcal{L}_n^1$.

Now, we have constructed N so that for N_D, $\delta_D([q_i], b) = [q_{i+1}]$ for $0 \leq i < n-1$ and $\delta_D([q_{n-1}], b) = d_\phi$, while $\delta_D(d_\phi, a) = d_\phi$, and hence $\mathcal{L}_n^2 = b^n a^*$ is a subset of the language $\mathcal{L}_n$ accepted by N.

Hence, in order to accept $\mathcal{L}_n$, after reading b^n, a state must have been reached where after every transition on a must result in an accept state, i.e. an XDFA state containing q_0. There is only one state that provides this possibility, and that is d_ϕ, which is excluded from the cycle needed to accept $\mathcal{L}_n^1$. Therefore, all 2^{n-1} odd-sized

states are necessary to accept $\mathcal{L}^1 \cup \mathcal{L}^2$. Let $\mathcal{L}_n$ be the language accepted by N, then since we have chosen $\mathcal{L}_n^1$ and $\mathcal{L}_n^2$ so that $\mathcal{L}_n^1 \cup \mathcal{L}_n^2 \subset \mathcal{L}_n$, at least 2^{n-1} states are necessary to accept $\mathcal{L}_n$. □

We illustrate Theorem 23 for $n = 4$.

Example 16. *Let $\phi(X) = X^3 + X + 1$, which is a primitive polynomial. Now, let N be an XNFA with transition matrices M_a and M_b. The matrix M_a is the normal form matrix of $c_a(X) = (X+1)\phi(X) = X^4 + X^3 + X^2 + 1$ and M_b the normal form matrix of $c_b(X) = X^4 + \phi(X) = X^4 + X^3 + X + 1$. Let $Q_0 = \{q_0\}$ and let $F^a = \{q_0\}$ and $F^r = \{q_1, q_2, q_3\}$. The matrices M_a and M_b are shown in Figures 4.16 and 4.17, while N and its equivalent XDFA N_D are shown in Figures 4.18 and 4.19. We have $\mathcal{L}^1 = a^{7i+j}$ for $i \geq 0$ and $j \in \{0, 4, 5\}$, $\mathcal{L}^2 = bbbba^*$ and $\mathcal{L}^1 \cup \mathcal{L}^2 \subset \mathcal{L}$. As Figure 4.19 shows, exactly $2^{n-1} = 2^{4-1} = 8$ states are necessary to represent $\mathcal{L}$ with an XDFA.*

$$M_a = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix}$$

Figure 4.16: Example 16: transition matrix for a

$$M_b = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \end{bmatrix}$$

Figure 4.17: Example 16: transition matrix for b

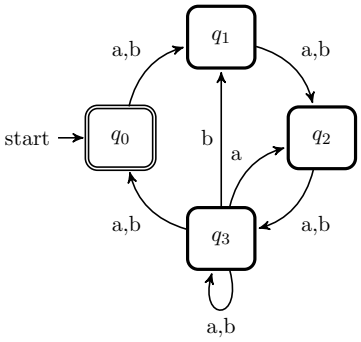


Figure 4.18: Example 16: N

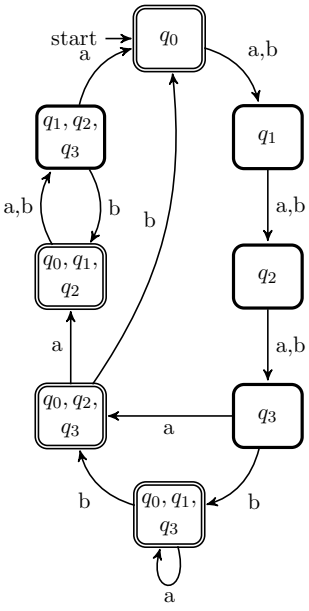


Figure 4.19: Example 16: N_D

□

The following is a simple corollary of Theorem 23.

Corollary 24. *For any m and any $n \geq 4$, there is a language $\mathcal{L}_n$ so that some n-state m-ary SV-XNFA accepts $\mathcal{L}_n$ and the minimal SV-XDFA that accepts $\mathcal{L}_n$ has 2^{n-1} states.*

We now show that any given SV-XNFA can be used to obtain another one via a so-called change of basis. Let $N = (Q, \Sigma, \delta, Q_0, F^a, F^r)$ an SV-XNFA with n states and transition matrices M_{σ_1}, M_{σ_2}, ..., M_{σ_r}, and let A be any non-singular $n \times n$ matrix. We encode Q_0 as a vector $v(Q_0)$ of length n over GF(2) and F^a and F^r as vectors $v(F^a)$ and $v(F^r)$, respectively. Let $N' = (Q, \Sigma, \delta', Q_0', F'^a, F'^r)$ be an SV-XNFA where $M'_{\sigma_i} = A^{-1}M_{\sigma_i}A$ for $0 \leq i \leq r$, $v(Q_0') = v(Q_0)A$, $v(F'^a)^T = A^{-1}v(F^a)^T$ and $v(F'^r)^T = A^{-1}v(F^r)^T$.

In the discussions in Sections 3.2 on page 12 and 3.5 on page 26 we showed that for unary and r-ary XNFA, if we apply a change of basis on an XNFA N to obtain N', then, $\Delta'(w) = \Delta(w)$. We showed in Section 4.1 on page 32 that for unary SV-XNFA, this holds for Δ^a and Δ^r, respectively the acceptance weight and rejection weight functions. We now show that this holds for r-ary SV-XNFA as well.

Let $w = \sigma_{i_1}\sigma_{i_2}\ldots\sigma_{i_k}$ be a word of length k, where σ_{i_j} represents the j-th letter of w. Then $\Delta^a(w) = v(Q_0)M_{\sigma_{i_1}}M_{\sigma_{i_2}}\ldots M_{\sigma_{i_k}}v(F^a)^T$, and

$$\begin{aligned}
\Delta'^a(w) &= v(Q_0')M'_{\sigma_{i_1}}M'_{\sigma_{i_2}}\ldots M'_{\sigma_{i_k}}v(F'^a)^T \\
&= (v(Q_0)A)(A^{-1}M_{\sigma_{i_1}}A)(A^{-1}M_{\sigma_{i_2}}A)\ldots(A^{-1}M_{\sigma_{i_k}}A)(A^{-1}v(F^a)^T) \\
&= v(Q_0)M_{\sigma_{i_1}}M_{\sigma_{i_2}}\ldots M_{\sigma_{i_k}}v(F^a)^T \\
&= \Delta^a(w)\ .
\end{aligned}$$

Similarly, $\Delta'^r(w) = \Delta^r(w)$. Clearly, the SV-condition is met by Δ'^a and Δ'^r, and so N' is an SV-XNFA that accepts the same language as N.

Finally, we give an example of such a change of basis for a trinary ($|\Sigma| = 3$) SV-XNFA.

Example 17. *Let N be an SV-XNFA with alphabet $\Sigma = \{a, b, c\}$, and the following transition matrices: M_a is the normal form matrix of $c_a(X) = X^4+X^3+X^2+1$, M_b is the normal form matrix of $c_b(X) = X^4+X^3+X+1$, and M_c is the normal form matrix of $c_c(X) = X^4+X^2+X+1$. Let $Q_0 = \{q_0\}$, $F^a = \{q_0, q_2\}$ and $F^r = \{q_1, q_3\}$. Figure 4.20 shows N and the equivalent XDFA N_D is given in Figure 4.21, where*

a double edge indicates an accept state and a thick edge indicates a reject state. Consider the following matrix A:

$$A = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} .$$

We use A to make a change of basis from N to N′. Let N′ be an XNFA with $\Sigma = \{a, b, c\}$*, where* $M'_a = A^{-1}M_aA$*,* $M'_b = A^{-1}M_bA$ *and* $M'_c = A^{-1}M_cA$*. Furthermore, let* $v(Q'_0) = v(Q_0)A$*, i.e.* $Q'_0 = \{q_1, q_2, q_3\}$*. Finally, let* $v(F'^a)^T = A^{-1}v(F^a)^T$ *and* $v(F'^r)^T = A^{-1}v(F^r)^T$*, i.e.* $F'^a = \{q_0, q_2\}$ *and* $F'^r = \{q_2, q_3\}$*. Figure 4.22, shows N′, with a double edge indicating an accept state, a thick edge indicating a reject state and a thick double edge indicating a state that is both an accept state and a reject state. Figure 4.23 gives the equivalent XDFA* N'_D*. It is worth noting that, although N′ has a different structure than N,* N'_D *has the same structure as* N_D*, and accepts the same language. Also, note that in* N'_D*, the state* $[q_0, q_1, q_2]$ *is a reject state, because it contains an even number of accept states, namely* q_0 *and* q_2*, but an odd number of reject states, namely* q_2*.* □

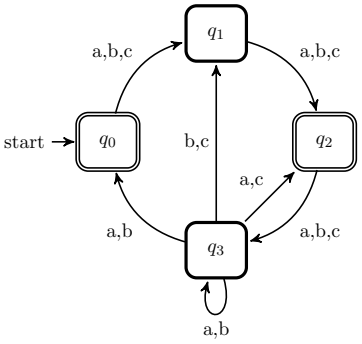


Figure 4.20: Example 17: N

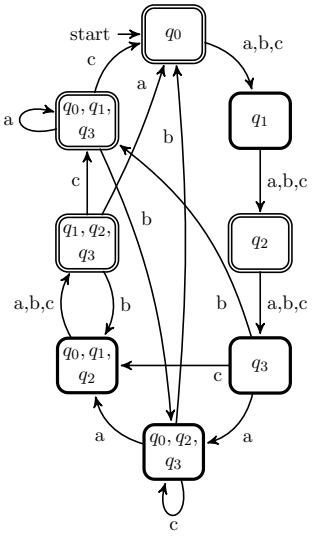


Figure 4.21: Example 17: N_D

4.4 Conclusion

In this chapter we presented self-verification as applied to XNFA. Self-verification is an instance of generalised acceptance, since there are two sets of final states, namely

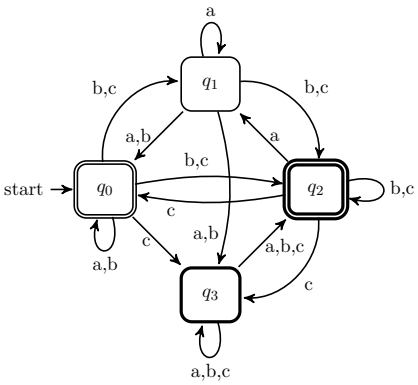


Figure 4.22: Example 17: N'

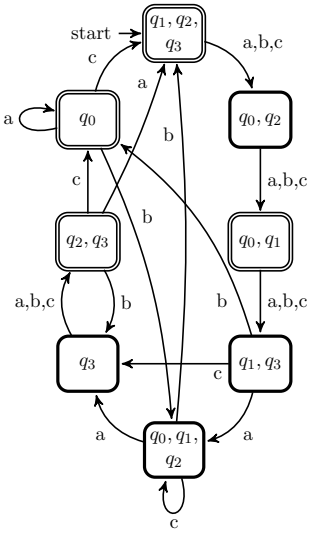


Figure 4.23: Example 17: N'_D

F^a and F^r, and specific meaning is associated with each, namely acceptance and rejection, respectively. The notion of self-verification as presented in the literature for typical NFA was interpreted for XNFA, retaining the so-called SV-requirement that any word be explicitly accepted or rejected by the automaton.

Because of the parity nature of acceptance for XNFA, an interesting difference between the SV-requirement for SV-NFA and SV-XNFA is the fact that $F^a \cap F^r$ must be empty for SV-NFA, but need not be for SV-XNFA. For SV-NFA, this follows from the fact that acceptance or rejection may be confirmed on the grounds of a single path ending in either kind of final state, which is similar to NFA, where acceptance is confirmed on the grounds of a single path ending in an accept state. This is a very intuitive requirement, since the idea of a state that both accepts and doesn't accept (or rejects, in the case of self-verification) seems to be a contradiction, and indeed it is a contradiction for NFA and SV-NFA.

However, for XNFA, acceptance requires that there be an odd number of paths ending in accept states, and hence, we have that for SV-XNFA, rejection requires an odd number of paths ending in reject states. This implies knowledge of all possible paths ending in either accept or reject states in order to count them, and hence, there is no reason to require that any single path contribute to only one of the counts. This requirement might additionally be imposed, as it is for disjunctive acceptance, but we have seen that the consequence is that the equivalence between XNFA and weighted automata over $GF(2)$ is removed.

The only requirement necessary in order to retain the equivalence with weighted automata over $GF(2)$ is that there be some paths which contribute to only one of the counts, so that ultimately, for every word, exactly an odd number of paths

contribute to the number of accept paths and an even number of paths contribute to the number of reject paths, or vice versa. We called this $GF(2)$-acceptance, and we studied this requirement primarily via the equivalent requirement for XDFA, which is that each XDFA state contain an odd number of XNFA states that belong to F^a and even number to F^r, or vice versa, in principle allowing XNFA states to belong to both. In reality then, while SV-NFA have three kinds of states, namely accept, reject, and neutral states, SV-XNFA with $GF(2)$-acceptance have four kinds of states, namely accept, reject, both, and neutral states.

Given this definition of SV-XNFA, we have shown that it is possible to obtain equivalent n-state SV-XNFA by performing a change of basis using some non-singular $n \times n$ matrix. We also note that the number of non-singular $n \times n$ matrices over GF(2) (including the identity matrix) is known to be $|GL(n, \mathbb{Z}_2)| = \prod_{k=0}^{n-1}(2^n - 2^k)$, and so, up to isomorphism, for any n-state SV-XNFA at most another $|GL(n, \mathbb{Z}_2)| - 1$ equivalent SV-XNFA can be found.

Chapter 5

Succinct representation of unary regular languages with $\star$-XNFA

XNFA have been shown to provide representations of certain unary and binary language families that are more succinct than what is achievable by NFA [32; 34]. For the unary case specifically, the family of languages that can be represented by n-state XNFA, but which requires $2^n - 1$ DFA states, are exactly those languages whose transition matrices have a primitive characteristic polynomial [32], and hence lead to cycles of length $2^n - 1$ in the equivalent minimal DFA. It is known that such cycles have good equidistribution properties [17]. We may say that a unary language $\mathcal{L}$ is equidistributed if the minimal XDFA that accepts it has good equidistribution properties. (This means that final and non-final states occur more or less equally frequently and are uniformly distributed over the cycle.) Hence, only equidistributed unary languages can be succinctly represented by XNFA with primitive characteristic polynomials. In this chapter we define $\star$-XNFA as an instance of generalised acceptance XNFA (GA-XNFA) (see Definition 2 on page 28) and investigate ways in which they can be used for the succinct representation of non-equidistributed unary languages.

Our particular focus is the descriptional complexity of the language operations intersection, union, relative complement and symmetric difference on unary languages, all of which lead to unary languages that are not equidistributed, except for symmetric difference. The main result is to show that the upper bound for the operations of intersection, union and relative complement is $(2^m - 1)(2^n - 1)$ on two languages represented minimally by XNFA with m and n states, respectively, and that the lower bound of $m + n$ is not tight. Furthermore, the same languages can be represented with unary $\star$-XNFA using only $m + n$ states. On the other hand, we show that the bound for symmetric difference is $m + n$ for both a unary XNFA and a unary $\star$-XNFA, and hence, given two languages that can be represented succinctly

Table 5.1: Example 18: transition function δ

δ	a	b
q_0	$\{q_1\}$	$\{q_1\}$
q_1	$\{q_2\}$	$\{q_2\}$
q_2	$\{q_0, q_1, q_2\}$	$\{q_0, q_2\}$

by two unary XNFA, their symmetric difference can also be represented succinctly by a unary XNFA.

The structure of this chapter is as follows: after defining $\star$-XNFA, we present some properties of strings in Section 5.1 that will be useful in proving state complexity bounds for intersection, union and relative complement in Section 5.2. Then we prove state complexity bounds for these operations and symmetric difference for $\star$-XNFA in Section 5.3. We then use the result for $\oplus$-XNFA to prove the state complexity bound for symmetric difference XNFA in Section 5.4, as well as providing an alternative proof for the bound given for complement in [30].

In Section 5.5, we return briefly to the question of a lower bound for XNFA on the operations of intersection, union and relative complement, which we conjecture to be at least polynomial. Finally, Section 5.6 contains a short discussion of the results presented.

Definition 5. *A $\star$-XNFA is a GA-XNFA with $\mathcal{F} = \{F^1, F^2, \ldots, F^r\}$, where $\star$ is a left-associative set operation applied to the languages $\mathcal{L}^1, \mathcal{L}^2, \ldots, \mathcal{L}^r$ associated with $F^1, F^2, \ldots, F^r$, respectively. A word w is accepted by a $\star$-XNFA N, if $w \in \mathcal{L}^1 \star \mathcal{L}^2 \star \cdots \star \mathcal{L}^r$.*[1]

Example 18. *Let N be a 3-state binary $\cap$-XNFA with δ given in Table 5.1, accepting the language $\mathcal{L}$. Let $\mathcal{F} = \{F^1, F^2\}$, where $F^1 = \{q_0, q_2\}$ and $F^2 = \{q_1, q_2\}$. The $\cap$-XNFA N is shown in Figure 5.1, with states belonging to F^1 indicated by a double border and states belonging to F^2 indicated by a thick border. Note that q_2 belongs to both. The equivalent XDFA N_D is given in Figure 5.2.*

We see, for example, that aa is accepted with respect to both F^1 and F^2, and hence $aa \in \mathcal{L}^1 \cap \mathcal{L}^2$, so $aa \in \mathcal{L}$. On the other hand, aab is accepted with respect to F^2 and rejected with respect to F^1, and hence $aab \notin \mathcal{L}^1 \cap \mathcal{L}^2$, so $aab \notin \mathcal{L}$. □

[1] Our focus is on expressions of this form, but in principle, future work may involve studying more complex expressions.

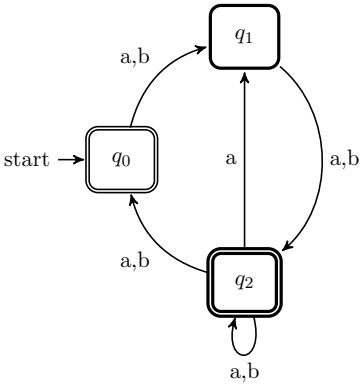


Figure 5.1: Example 18: N

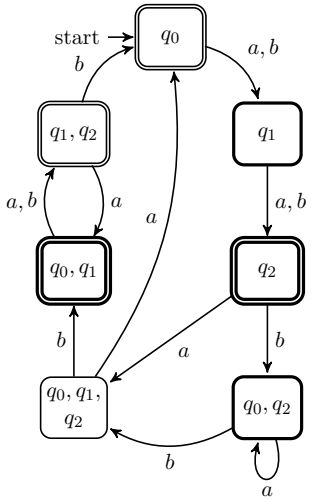


Figure 5.2: Example 18: N_D

5.1 Properties of binary strings: slices and permutations

A unary XNFA with a non-singular transition matrix has an equivalent minimal XDFA that consists of a single cycle of states (see Example 7 on page 22). If the cycle has length n, then the language $\mathcal{L}$ recognised by this XNFA and XDFA can be represented as a binary string s of length n, indexed from 0 to $n-1$, where a 1 in position i corresponds to reaching a final state after reading i characters, or $a^i \in \mathcal{L}$, and a 0 corresponds to reaching a non-final state, or $a^i \notin \mathcal{L}$. For example, the string 10111 represents the language $\mathcal{L} = a^{5i} + j$, for $j \in \{0, 2, 3, 4\}$ and $i \geq 0$. (Note that $a^0 = \epsilon$, which is in $\mathcal{L}$, since the element in position 0 is a 1.)

In order to prove the state complexity bound for the intersection, union and relative complement of unary languages for XNFA, we therefore first establish some useful properties of strings. We provide some definitions and lemmas relating to so-called slices of strings, before defining k-displacements and shifted k-displacements, which are permutations of strings, and we give some results concerning these displacements. This puts us in a position to prove Theorems 34, 35 and 36, which relate specifically to binary strings. We use these results in our discussion of the descriptional complexity of certain language operations for unary XNFA in Section 5.2.

5.1.1 Notation

In this section, we discuss strings of finite length. We use superscripts primarily to indicate repeated concatenation, so that $s = t^3 = ttt$; that is, s is t concatenated 3 times. In order to facilitate naming of strings, however, we may also use

superscripts to name related strings. In this case, we will define the strings with specific superscripts and use parentheses to distinguish from those superscripts that indicate repeated concatenation. For example, if we define a string s^1, we indicate a repeated concatenation of s^1 with $(s^1)^x$. In this chapter, the intended meaning of a superscript will be clear from the context.

We indicate the elements of strings using subscripts starting at 0, and so s_0 is the first element in a string and s_{n-1} is the final element in a string of length n. We say s_i is the element or value of s at index i.

Note that for any $s = (s')^x$ of length n where s' has length d, it follows that $s_i = s'_{i \bmod d}$ for $0 \leq i \leq n-1$.

5.1.2 Properties of strings

We start with some basic definitions.

Definition 6. *[19] Let s be a string and let t be the shortest string such that $s = t^x$ for some x, where t has length p. Then s has period p.*

Definition 7. *[19] A primitive string is a string s whose period equals its length.*

Lemma 25. *Let $y \geq 1$. The strings s and s^y have the same period.*

Proof. If $s = t^x$ for some primitive string t, then $s^y = (t^x)^y = t^{xy}$. It follows that s and s^y have the same period. □

Definition 8. *Let $s = s_0 s_1 \cdots s_{n-1}$ be a string of length n. Then*

$$s' = s_{q \bmod n} s_{1+q \bmod n} \cdots s_{(n-1+q) \bmod n}$$

is a shift of s. Clearly, the period of s' is equal to the period of s.

Example 19. *Let $s = 100111$, which has length and period 6. Then we can list all its shifts as follows:*

$$\{001111, 011110, 111100, 111001, 110011\} .$$

Evidently, each of these strings also have period 6. On the other hand, let $s' = 101101$, which has length 6 and period 3. Then we can list all its shifts as follows:

$$\{011011, 110110, 101101, 011011, 110110\} .$$

Note that, in addition to all these shifts having period 3, some shifts are identical to others. □

Example 20. *Let* $s = 111001110011100$*, so that* $n = 15$*. Let* $t = 11100$*, which is primitive, since it has length and period of* 5*. Then* $s = (11100)^3 = t^3$*, and hence* s *has period* 5*. Now, let* $s' = 100111001110011$ *and let* $t' = 10011$*. Clearly,* t' *is a rightward shift of* t *by three positions, and* t' *has period* 5*. Furthermore,* $s' = (10011)^3 = (t')^3$*, so* s' *also has period* 5*. Finally,* s' *is a rightward shift of* s *by three positions.* □

The following definition introduces the notion of q-slices of strings, which will be useful when reasoning about the periods of certain strings.

Definition 9. *Let* $s = s_0 s_1 \cdots s_{n-1}$ *be a string of length* n *and let* $dk = n$*. Now, let*

$$\chi_d(s, q) = s_{0+(q \bmod d)} s_{d+(q \bmod d)} s_{2d+(q \bmod d)} \cdots s_{(k-1)d+(q \bmod d)} \ .$$

We say that $\chi_d(s, q)$ *is the* q*-slice of* s *with respect to* d*. The length of* $\chi_d(s, q)$ *is* $k = \frac{n}{d}$*.*

Note that if $q < d$, we can write

$$\chi_d(s, q) = s_{0+q} s_{d+q} s_{2d+q} \cdots s_{(k-1)d+q} \ .$$

This observation leads to the following lemma.

Lemma 26. *Let* $s = s_0 s_1 \cdots s_{n-1}$ *be a string of length* n*. Then for any* q *and* $q' = q \bmod d$*,*

$$\chi_d(s, q) = \chi_d(s, q') \ .$$

Proof. We simply note that

$$\begin{aligned} \chi_d(s, q) &= s_{0+(q \bmod d)} s_{d+(q \bmod d)} s_{2d+(q \bmod d)} \cdots s_{(k-1)d+(q \bmod d)} \\ &= s_{0+q'} s_{d+q'} s_{2d+q'} \cdots s_{(k-1)d+q'} \\ &= \chi_d(s, q') \ . \end{aligned}$$

□

Essentially, a q-slice with respect to d divides the string into equal parts of length d and constructs a new string by taking the q-th element from each part.

Example 21. *Let* $s_1 = 101101101$*,* $s_2 = 101101011$ *and let* $s_3 = 110111101111011$*, so that* $n_1 = n_2 = 9$ *and* $n_3 = 15$*. Some examples of* q*-slices are as follows:*

$$\chi_3(s_1, 1) = 000 \qquad \chi_3(s_3, 1) = 11011$$
$$\chi_3(s_1, 2) = 111 \qquad \chi_3(s_3, 2) = 01111$$
$$\chi_3(s_2, 1) = 001 \qquad \chi_5(s_3, 2) = 000$$
$$\chi_3(s_2, 2) = 111 \qquad \chi_5(s_3, 4) = 111\ .$$

Figure 5.3 illustrates how $\chi_3(s_1, 1)$ is found, while Figures 5.4 and 5.5 illustrate the same for $\chi_3(s_2, 1)$ and $\chi_3(s_3, 2)$, respectively. The positions of the dashed lines that divide the strings into sections of equal length are determined by the value of d, while the position of the blocks within each section is a consequence of the choice of q.

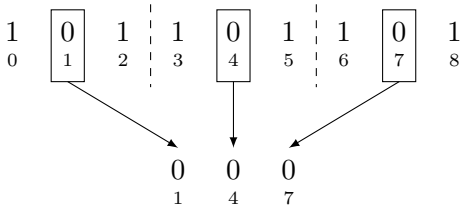


Figure 5.3: Example 21: $\chi_3(s_1, 1)$, with $d = 3$ and $q = 1$

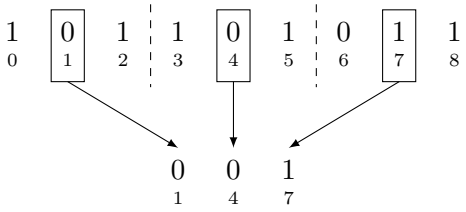


Figure 5.4: Example 21: $\chi_3(s_2, 1)$, with $d = 3$ and $q = 1$

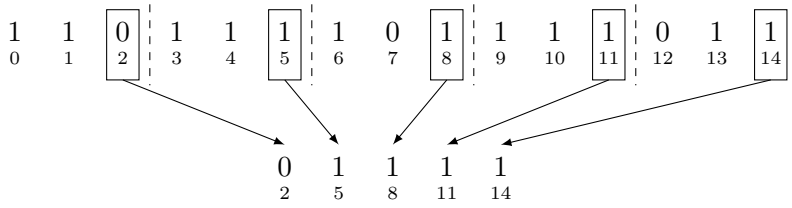


Figure 5.5: Example 21: $\chi_3(s_3, 2)$, with $d = 3$ and $q = 2$

□

We now give some lemmas regarding relationships that exist between the periods of q-slices of related strings.

Lemma 27. *Let t be a string of length p and let $df = p$. Then,*

$$\chi_d(t^r, q) = \chi_d(t, q)^r \ .$$

Proof. Since $\chi_d(t^r, q) = \chi_d(t^r, q')$ for some $q' < d$ by Lemma 26, we may assume, without loss of generality, that $q < d$. Let $t = t_0 t_1 \cdots t_{p-1}$. Then

$$\begin{aligned} t^r &= (t_0 t_1 \cdots t_{p-1})^r \\ &= \underbrace{(t_0 t_1 \cdots t_{p-1})(t_0 t_1 \cdots t_{p-1}) \cdots (t_0 t_1 \cdots t_{p-1})}_{r \text{ times}} \ . \end{aligned}$$

Also,

$$\chi_d(t, q) = (t_{0+q} t_{d+q} t_{2d+q} \cdots t_{(f-1)d+q}) \ .$$

Let n be the length of t^r, and hence $pr = n$. From Definition 9, we know that the length of $\chi_d(t, q)$ is $\frac{p}{d} = f$. Then we have,

$$\begin{aligned} \chi_d(t^r, q) &= \underbrace{(t_{0+q} t_{d+q} t_{2d+q} \cdots t_{(f-1)d+q}) \cdots (t_{0+q} t_{d+q} t_{2d+q} \cdots t_{(f-1)d+q})}_{r \text{ times}} \\ &= (t_{0+q} t_{d+q} t_{2d+q} \cdots t_{(f-1)d+q})^r \\ &= \chi_d(t, q)^r \ . \end{aligned}$$

□

Lemma 28. *If a string s has period p, then for any q, the q-slice of s with respect to p has period 1. Conversely, if any q-slice with respect to d of a string s has period 1, then s has period p such that $p \mid d$.*

Proof. Let s have period p. Then there is some string $t = t_0 t_1 \cdots t_{p-1}$ of length p such that $s = t^x$ for some x. Let $0 \leq q \leq p - 1$, then

$$\begin{aligned} \chi_p(s, q) &= s_q s_{p+q} s_{2p+q} \cdots s_{(x-1)p+q} \\ &= t_{q \bmod p} t_{(p+q) \bmod p} t_{(2p+q) \bmod p} \cdots t_{[(x-1)p+q] \bmod p} \\ &= \underbrace{t_q t_q t_q \cdots t_q}_{x \text{ times}} \ , \end{aligned}$$

which clearly has period 1.

Conversely, let $\chi_d(s, q)$ have period 1 for any q. By Lemma 26, we need only consider $0 \leq q \leq d - 1$. Then it follows that, for any $0 \leq q \leq d - 1$,

$$s_q = s_{d+q} = s_{2d+q} = \cdots = s_{(\frac{n}{d}-1)d+q} \ .$$

Let $w = (s_0 s_1 ... s_{d-1})$, then we have $s = (s_0 s_1 ... s_{d-1})^{\frac{n}{d}} = w^{\frac{n}{d}}$. Let w have period p, then $p \mid d$, and by Lemma 25, s has period p. □

We illustrate Lemma 28 in the next example.

Example 22. *Let* $s = 101101101101$ *so that* $n = 12$ *and* $p = 3$*; that is* $s = (101)^4$*. Then we have*

$$\chi_3(s, 0) = 1111$$
$$\chi_3(s, 1) = 0000$$
$$\chi_3(s, 2) = 1111 \ .$$

In all cases, the period of $\chi_3(s, q)$ *is* 1*. Figure 5.6 illustrates how, for example,* $\chi_3(s, 0)$ *is found.*

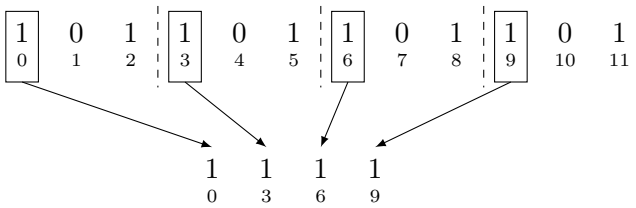


Figure 5.6: Example 22: $\chi_3(s, 0)$, with $d = 3$ and $q = 0$

□

Lemma 29. *If a string* s *of length* n *has period* p *and if* $df = p$*, then for any* q*, the* q*-slice of* s *with respect to* d *has period* f' *such that* $f' \mid f$*.*

Proof. Since s has period p, there is some primitive string $t = t_0 t_1 \cdots t_{p-1}$ of length p such that $s = t^{\frac{n}{p}}$. Let $dk = n$ and let $0 \leq q \leq d - 1$, then

$$\chi_d(s, q) = s_{0+q} s_{d+q} s_{2d+q} \cdots s_{(k-1)d+q},$$

which has length $\frac{n}{d} = k$. But $s = t^{\frac{n}{p}}$, so by Lemma 27,

$$\begin{aligned} \chi_d(s, q) &= \chi_d(t^{\frac{n}{p}}, q) \\ &= \chi_d(t, q)^{\frac{n}{p}} \ . \end{aligned}$$

By Lemma 25, the period of $\chi_d(s,q)$ is equal to the period of $\chi_d(t,q)$. We know that the length of $\chi_d(s,q)$ is $\frac{n}{d} = k$, and hence the length of $\chi_d(t,q)^x$ must also be k. Let the length of $\chi_d(t,q)$ be y, then $(\frac{n}{p})(y) = k$, and so

$$y = \frac{pk}{n} = \frac{p}{d} = f \ .$$

Therefore, the length of $\chi_d(t,q)$ is f, and hence the period of $\chi_d(t,q)$ – and therefore the period of $\chi_d(s,q)$ – must be some f' such that $f' \mid f$. □

Example 23. *Let* $s = (110000111011)^3$*, so that* $n = 36$ *and* $p = 12$*. Let* $d = 2$ *and* $f = 6$*, so that* $df = p$*. Then we have,*

$$\chi_2(s,0) = (100111)^3$$
$$\chi_2(s,1) = (100101)^3 \ .$$

In both cases, the period is 6*, which divides* $f = 6$*.* □

We now introduce the notion of k-displacements of a string, which are a specific kind of permutation of a string which results in an ordered rearrangement of the string.

Definition 10. *Let* $\gcd(k,n) = 1$ *and* $\kappa(i,k) = i \cdot k \mod n$*, and let* s *be a string of length* n*. A* k*-displacement of* s *is a permutation* s' *of* s *where the value at index* i *in* s' *is the value at index* $\kappa(i,k)$ *in* s*.*

Example 24. *Let* $s = 011100011$*, so that* $n = 9$*. We may represent the indices of* s *as* $(0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8)$*. Now let* $k = 4$*, and let* s' *be the 4-displacement of* s*, so that* $\kappa(i,4) = (i \cdot 4) \mod 9$*. We compute the value for* s' *at each index by determining the corresponding index of* s*:*

$$\begin{aligned}
\kappa(0,4) &= (0 \cdot 4) \bmod 9 &&= 0\\
\kappa(1,4) &= (1 \cdot 4) \bmod 9 &&= 4\\
\kappa(2,4) &= (2 \cdot 4) \bmod 9 &&= 8\\
\kappa(3,4) &= (3 \cdot 4) \bmod 9 &&= 3\\
\kappa(4,4) &= (4 \cdot 4) \bmod 9 &&= 7\\
\kappa(5,4) &= (5 \cdot 4) \bmod 9 &&= 2\\
\kappa(6,4) &= (6 \cdot 4) \bmod 9 &&= 6\\
\kappa(7,4) &= (7 \cdot 4) \bmod 9 &&= 1\\
\kappa(8,4) &= (8 \cdot 4) \bmod 9 &&= 5
\end{aligned}$$

Hence, s' *corresponds to the ordered arrangement* $(0\ 4\ 8\ 3\ 7\ 2\ 6\ 1\ 5)$ *of the indices of* s*, and so* $s' = 001111010$*.*

Figure 5.7 illustrates this visually. Since we compute κ modulo 9, which is the length of s, we can think of s as extended as long as necessary. At the top of the figure, therefore, we have repeated instances of s, separated by dashed lines, with the corresponding index indicated below each value. Note that in the representation of s' below, the difference modulo 9 between adjacent indices is 4.

From the figure it is evident that we can think of k-displacements as slices of extended strings, where d has the value of k and q is fixed at 0. We call them k-displacements to emphasise the fact that the resulting string is a permutation of the original, with the indices displaced so that the difference modulo n between adjacent indices is k.

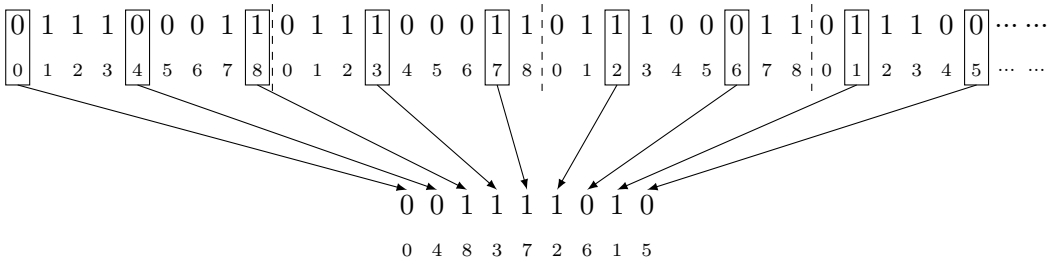


Figure 5.7: Example 24: $\kappa(i, 4) = (i \cdot 4) \mod 9$

□

A shifted k-displacement is similar to a k-displacement, but the original string is shifted before the ordered rearrangement is computed.

Definition 11. *Let s be a string of length n. A shifted k-displacement of s is a permutation s_k of s where the value at index i in s_k is the value at index $\kappa_q(i, k)$ in s, where $\gcd(k, n) = 1$ and $\kappa_q(i, k) = (i \cdot k \ + q) \bmod n$ for some q.*

Since $(i \cdot k \ + q) \bmod n = [i \cdot k \bmod n + q \bmod n] \bmod n$, a shifted k-displacement is a shift of a k-displacement.

Example 25. *Let $s = 011100011$ so that $n = 9$. We may represent the indices of s as $(0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8)$. Now let $k = 4$ and $q = 2$, and let s' be a 2-shifted 4-displacement of s. We compute the value for s'' at each index by determining the*

corresponding index of s:

$$\begin{aligned}
\kappa_2(0,4) &= (0 \cdot 4 + 2) \bmod 9 &&= 2\\
\kappa_2(1,4) &= (1 \cdot 4 + 2) \bmod 9 &&= 6\\
\kappa_2(2,4) &= (2 \cdot 4 + 2) \bmod 9 &&= 1\\
\kappa_2(3,4) &= (3 \cdot 4 + 2) \bmod 9 &&= 5\\
\kappa_2(4,4) &= (4 \cdot 4 + 2) \bmod 9 &&= 0\\
\kappa_2(5,4) &= (5 \cdot 4 + 2) \bmod 9 &&= 4\\
\kappa_2(6,4) &= (6 \cdot 4 + 2) \bmod 9 &&= 8\\
\kappa_2(7,4) &= (7 \cdot 4 + 2) \bmod 9 &&= 3\\
\kappa_2(8,4) &= (8 \cdot 4 + 2) \bmod 9 &&= 7
\end{aligned}$$

Hence, s'' corresponds to the ordered arrangement (2 6 1 5 0 4 8 3 7) *of the indices of s, and so $s'' = 010001111$, as illustrated in Figure 5.8. Note that s'' is a shift of s' in Example 24, whose 0-th index corresponds to the 2-nd (or q-th) index of s. The dotted rectangles and arrows in the figure indicate the shift caused by q.*

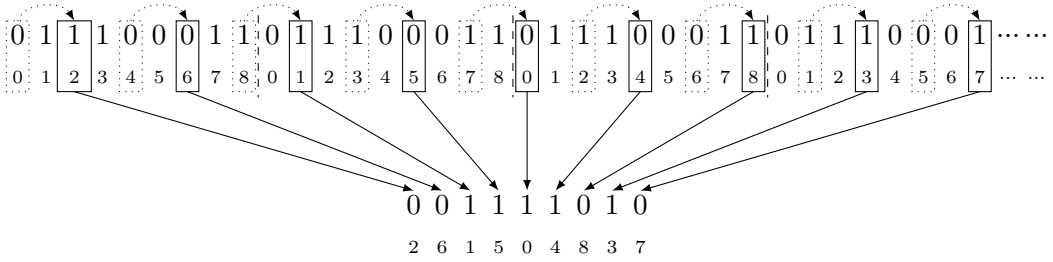


Figure 5.8: Example 24: $\kappa_2(i, 4) = (i \cdot 4 \ + 2) \bmod 9$

□

The following lemma is concerned with the relation between indices of a string s and the indices of any repeating substring w of s.

Lemma 30. *Let $q \mid n$ and let s be a string of length n and w a string of length q such that $s = w^{\frac{n}{q}}$. Then for any $0 \leq i, j \leq n-1$, if $i \bmod q = j \bmod q$, then $s_i = s_j$. Conversely, let s be a string of length n, where for some q, it is true that $s_i = s_j$ for any $0 \leq i, j \leq n-1$ such that $i \bmod q = j \bmod q$. Then s has a substring w of length q such that $s = w^{\frac{n}{q}}$.*

Proof. First, let $s = w^{\frac{n}{q}}$, and let $w = w_0 w_1 \cdots w_{q-1}$. Then $s = (w_0 w_1 \cdots w_{q-1})^{\frac{n}{q}}$, and hence $s_i = w_{i \bmod q}$ and $s_j = w_{j \bmod q}$. Hence, for any $0 \leq i, j \leq n-1$, if $i \bmod q = j \bmod q$, then $s_i = w_{i \bmod q} = w_{j \bmod q} = s_j$.

Conversely, let s be a string of length n, where for some q, it is true that $s_i = s_j$ for any $0 \leq i, j \leq n-1$ such that $i \bmod q = j \bmod q$. Then, for $0 \leq k \leq q-1$, it follows that $s_k = s_{q+k} = s_{2q+k} = \cdots = s_{(\frac{n}{q}-1)q+k}$, and hence $s = (s_0 s_1 \cdots s_{q-1})^{\frac{n}{q}}$. Let $w = (s_0 s_1 \cdots s_{q-1})$, then $s = w^{\frac{n}{q}}$. □

The following lemma relates to the period of k-displacements and shifted k-displacements.

Lemma 31. *The period of a string is equal to the period of any k-displacement or shifted k-displacement of that string.*

Proof. Let $s = s_0 s_1 \cdots s_{n-1}$ be a string of length n and let p be its period. Then we can write $s = t^{\frac{n}{p}}$, where t is a string of length p. Hence, by Lemma 30, the values at any indices i and j of s are equal if $i \bmod p = j \bmod p$. The value at any index r of a k-displacement s' of s is equal to the value of s at $\kappa(r, k)$. Without loss of generality, we can assume that $k < n$. Now, since $p \mid n$, it follows that $i \bmod n = xp + (i \bmod p)$ for some x. Hence, for $0 \leq i, j \leq n-1$, if $i \bmod p = j \bmod p$, then

$$\begin{aligned}
\kappa(i,k) \bmod p &= (i \cdot k \bmod n) \bmod p \\
&= ([(i \bmod n)(k \bmod n)] \bmod n) \bmod p \\
&= ([(xp + (i \bmod p))(k \bmod n)] \bmod n) \bmod p \\
&= ([(xp + (j \bmod p))(k \bmod n)] \bmod n) \bmod p \\
&= ([(j \bmod n)(k \bmod n)] \bmod n) \bmod p \\
&= (j \cdot k \bmod n) \bmod p \\
&= \kappa(j,k) \bmod p\ .
\end{aligned}$$

That is, if $i \bmod p = j \bmod p$, then $\kappa(i,k) \bmod p = \kappa(j,k) \bmod p$ as long as $\gcd(k,n) = 1$. But $\kappa(i,k)$ and $\kappa(j,k)$ are indices for which the values of s are equal to the values of s' at i and j, respectively. Hence, for $0 \leq i, j \leq n-1$, if $i \bmod p = j \bmod p$, then $s'_i = s'_j$.

Therefore, by Lemma 30, $s' = w^{\frac{n}{p}}$, where w has length p. By Lemma 25, the period of s' is equal to the period p' of w, where $p' \mid p$. Stated generally, the period of a k-displacement of a string divides the period of the string.

Recall that k is chosen so that $\gcd(n,k) = 1$, and therefore it has a modular multiplicative inverse[2] k^{-1} so that $(k \cdot k^{-1}) \bmod n = 1$. Let s'' be a k^{-1}-displacement of s'. That is, given some index r of s'', its value is equal to the value of s' at $\kappa(r, k^{-1})$. Furthermore, given any index $\kappa(r, k^{-1})$ of s', its value is equal to the value of s at $\kappa(\kappa(r, k^{-1}), k)$. However,

[2]Given $\mathbb{Z}_n$, namely the integers modulo n, any k has a multiplicative inverse k^{-1} if and only if $\gcd(n,k) = 1$. Then $(k \cdot k^{-1}) \bmod n = 1$.

$$\begin{aligned}
\kappa(\kappa(r, k^{-1}), k) &= \kappa([r \cdot k^{-1} \bmod n], k) \\
&= (r \cdot k^{-1} \bmod n)k \bmod n \\
&= (r \cdot k^{-1} \bmod n)(k \bmod n) \bmod n \\
&= (r \cdot k^{-1} \cdot k) \bmod n \\
&= (r \bmod n)(k^{-1} \cdot k \bmod n) \bmod n \\
&= r \bmod n \\
&= r
\end{aligned}$$

Hence, $s'' = s$, and so s is a k^{-1}-displacement of s'. Therefore, the period p of s must divide p'. Since $p' \mid p$ and $p \mid p'$, it follows that $p' = p$, and hence the period of s' is p. Furthermore, shifting a string does not affect its period, therefore the same is true for any s' that is a shifted k-displacement of s. □

Example 26. *From Examples 24 and 25, we have* $s = 011100011$*, its* 4*-displacement* $s' = 001111010$*, and its* 3*-shifted* 4*-displacement* $s'' = 010001111$*. We see that* s*,* s' *and* s'' *have period* 9*.*

Let $r = 110110110 = (110)^3$ *with length* 9 *and period* 3*. Let* $k = 2$ *and* $q = 5$*, so that the* 2*-displacement of* r *is* $r' = 101101101 = (101)^3$*, and its* 5*-shifted* 2*-displacement is* $r'' = 011011011 = (011)^3$*. Then* r*,* r' *and* r'' *all have equal period* 3*. The ordered arrangement of indices for* r' *is computed as follows:*

$$\begin{aligned}
\kappa(0, 2) &= (0 \cdot 2) \bmod 9 &&= 0 \\
\kappa(1, 2) &= (1 \cdot 2) \bmod 9 &&= 2 \\
\kappa(2, 2) &= (2 \cdot 2) \bmod 9 &&= 4 \\
\kappa(3, 2) &= (3 \cdot 2) \bmod 9 &&= 6 \\
\kappa(4, 2) &= (4 \cdot 2) \bmod 9 &&= 8 \\
\kappa(5, 2) &= (5 \cdot 2) \bmod 9 &&= 1 \\
\kappa(6, 2) &= (6 \cdot 2) \bmod 9 &&= 3 \\
\kappa(7, 2) &= (7 \cdot 2) \bmod 9 &&= 5 \\
\kappa(8, 2) &= (8 \cdot 2) \bmod 9 &&= 7
\end{aligned}$$

The ordered arrangement of indices for r'' is computed as follows:

$$\begin{aligned}
\kappa_5(0,2) &= (0 \cdot 2 + 5) \bmod 9 &&= 5\\
\kappa_5(1,2) &= (1 \cdot 2 + 5) \bmod 9 &&= 7\\
\kappa_5(2,2) &= (2 \cdot 2 + 5) \bmod 9 &&= 0\\
\kappa_5(3,2) &= (3 \cdot 2 + 5) \bmod 9 &&= 2\\
\kappa_5(4,2) &= (4 \cdot 2 + 5) \bmod 9 &&= 4\\
\kappa_5(5,2) &= (5 \cdot 2 + 5) \bmod 9 &&= 6\\
\kappa_5(6,2) &= (6 \cdot 2 + 5) \bmod 9 &&= 8\\
\kappa_5(7,2) &= (7 \cdot 2 + 5) \bmod 9 &&= 1\\
\kappa_5(8,2) &= (8 \cdot 2 + 5) \bmod 9 &&= 3
\end{aligned}$$

□

Finally, the following lemma shows a specific relationship between q-slices and k-displacements.

Lemma 32. *Let $s = t^k$ be a string of length pk with period p, where $\gcd(p,k) = 1$. Let $p = p_\alpha p_\beta \neq 1$ and $k = k_\alpha k_\beta \neq 1$. Then*

$$\chi_{p_\alpha k_\alpha}(s,q) = (t')^{k_\beta}$$

where t' is a k_α-displacement of $\chi_{p_\alpha}(t,q)$.

Proof. We have

$$\chi_{p_\alpha}(t,q) = t_q t_{p_\alpha+q} t_{2p_\alpha+q} \cdots t_{(p_\beta-1)p_\alpha+q} \ .$$

Then the following is a k_α-displacement of $\chi_{p_\alpha}(t,q)$, since every index has the form $i \cdot k \bmod n$:

$$t' = t_{q \bmod p} t_{(p_\alpha k_\alpha+q) \bmod p} t_{(2p_\alpha k_\alpha+q) \bmod p} \cdots t_{[(p_\beta-1)p_\alpha k_\alpha+q] \bmod p} \ .$$

Furthermore, we have

$$\begin{aligned}
&[(p_\beta - 1)p_\alpha k_\alpha + q] \bmod p\\
=&([(p_\beta - 1)p_\alpha k_\alpha] \bmod p + [q \bmod p]) \bmod p\\
=&([p - p_\alpha k_\alpha] \bmod p + [q \bmod p]) \bmod p\\
=&[(-p_\alpha k_\alpha) \bmod p + (q \bmod p)] \bmod p\\
=&(-p_\alpha k_\alpha + q) \bmod p \ ,
\end{aligned}$$

and also

$$\begin{aligned}
&[(rp_\beta - 1)p_\alpha k_\alpha + q] \bmod p \\
=&([(rp_\beta - 1)p_\alpha k_\alpha] \bmod p + [q \bmod p]) \bmod p \\
=&([rp - p_\alpha k_\alpha] \bmod p + [q \bmod p]) \bmod p \\
=&[(-p_\alpha k_\alpha) \bmod p + (q \bmod p)] \bmod p \\
=&(-p_\alpha k_\alpha + q) \bmod p \ .
\end{aligned}$$

Hence,

$$[(rp_\beta - 1)p_\alpha k_\alpha + q] \bmod p = [(p_\beta - 1)p_\alpha k_\alpha + q] \bmod p \ .$$

Consequently,

$$\begin{aligned}
\chi_{p_\alpha k_\alpha}(s, q) &= s_q s_{p_\alpha k_\alpha + q} s_{2p_\alpha k_\alpha + q} \cdots s_{(p_\beta k_\beta - 1)p_\alpha k_\alpha + q} \\
&= \left(s_q s_{p_\alpha k_\alpha + q} s_{2p_\alpha k_\alpha + q} \cdots s_{(p_\beta - 1)p_\alpha k_\alpha + q}\right)^{k_\beta} \\
&= \left(t_{q \bmod p} t_{(p_\alpha k_\alpha + q) \bmod p} t_{(2p_\alpha k_\alpha + q) \bmod p} \cdots t_{[(p_\beta - 1)p_\alpha k_\alpha + q] \bmod p}\right)^{k_\beta} \\
&= (t')^{k_\beta} \ .
\end{aligned}$$

□

5.1.3 Bitwise binary operations on strings

Definition 12. *Let $s^1 = s^1_0 s^1_1 \dots s^1_{n-1}$ and $s^2 = s^2_0 s^2_1 \dots s^2_{n-1}$ be two binary strings of length n. Then $s^- = s^-_0 s^-_1 \dots s^-_{n-1}$, where $s^-_i = 1$ if $s^1_i = 1$ and $s^2_i = 0$, and 0 otherwise. We say that s^- is the result of performing a bitwise DIFF operation on s^1 and s^2.*

The bitwise DIFF operation is analogous to the relative complement (or difference) set operation, which is defined as $A \setminus B$ for any sets A and B. Just as $A \setminus B$ contains only those elements of A that do not also occur in B, s^- contains only 1's in positions where a 1 occurs in s^1 but not also in s^2.

The following lemma will enable us to show that the proof of Theorem 34 is similar to those of Theorems 35 and 36.

Lemma 33. *Let s^1 and s^2 be two binary strings of length n. Let $s^\star = s^1 \star s^2$, where $\star$ is either the bitwise AND operator, the bitwise OR operator or the bitwise DIFF operator. Then*

AND. *If $s^1 = 1^n$, $s^\wedge = 1^n$ if and only if $s^2 = 1^n$, and $s^\wedge = 0^n$ if and only if $s^2 = 0^n$. Furthermore, the period of $s^\wedge$ is equal to the period of s^2.*

OR. *If $s^1 = 0^n$, $s^\vee = 0^n$ if and only if $s^2 = 0^n$, and $s^\vee = 1^n$ if and only if $s^2 = 1^n$. Furthermore, the period of $s^\vee$ is equal to the period of s^2.*

DIFF. *If* $s^1 = 1^n$, $s^- = 0^n$ *if and only if* $s^2 = 1^n$, *and* $s^- = 1^n$ *if and only if* $s^2 = 0^n$. *Furthermore, the period of* s^- *is equal to the period of* s^2.

Proof. The lemma follows directly from the definitions of bitwise AND, bitwise OR and bitwise DIFF. □

We give the following theorem and then demonstrate one of the cases addressed in the proof in a subsequent example.

Theorem 34. *Let* s^1 *and* s^2 *be two primitive binary strings of length* m *and* n, *respectively, where* $\gcd(m, n) = 1$ *and hence* $\operatorname{lcm}(m, n) = mn$. *Let* $s^{1\prime} = (s^1)^n$ *and* $s^{2\prime} = (s^2)^m$, *so that* $s^{1\prime}$ *and* $s^{2\prime}$ *are strings of length* mn. *Now, let* $s^\wedge = s^{1\prime} \wedge s^{2\prime}$, *i.e.* $s^\wedge$ *is the result of performing a bitwise AND operation on* $s^{1\prime}$ *and* $s^{2\prime}$. *Then* $s^\wedge$ *is a primitive string.*

Proof. Since the theorem concerns the bitwise AND operation between two strings, we are specifically interested in those positions where a 1 occurs in both strings, since these are the only pairs of bits that will result in a 1 in $s^\wedge$.

The length of each of the strings $s^{1\prime}$, $s^{2\prime}$ and $s^\wedge$ is mn and hence the period of $s^\wedge$ must be such that it divides mn. We identify four exhaustive cases for possible divisors of mn and prove that $s^\wedge$ can only have period mn.

Case 1: $d = m$ (or $d = n$). By Lemma 25 on page 59, we know that $s^{1\prime}$ has period m. Then by Lemma 28 on page 62, $\chi_m(s^{1\prime}, q)$ has period 1 for any $0 \leq q \leq m - 1$. Let q be some index of s^1 such that $s^1_q = 1$. Then $\chi_m(s^{1\prime}, q) = 1^n$.

We also have

$$\begin{aligned}\chi_m(s^{2\prime}, q) &= s^{2\prime}_q s^{2\prime}_{m+q} s^{2\prime}_{2m+q} \cdots s^{2\prime}_{(n-1)m+q} \\ &= s^2_{q \bmod n} s^2_{(m+q) \bmod n} s^2_{(2m+q) \bmod n} \cdots s^2_{[(n-1)m+q] \bmod n} .\end{aligned}$$

Since $\gcd(m, n) = 1$, $\chi_m(s^{2\prime}, q)$ is a shifted m-displacement of s^2. Therefore, by Lemma 31, the period of $\chi_m(s^{2\prime}, q)$ is equal to the period of s^2, which is n. Now, $\chi_m(s^\wedge, q) = \chi_m(s^{1\prime}, q) \wedge \chi_m(s^{2\prime}, q)$. Since $\chi_m(s^{1\prime}, q) = 1^n$ and $\chi_m(s^{2\prime}, q)$ has period n, by Lemma 33, the period of $\chi_m(s^\wedge, q)$ is n and not 1. Therefore, by Lemma 28, $s^\wedge$ cannot have period m. Similarly, $s^\wedge$ cannot have period n.

Case 2: d such that $d \mid m$ (or $d \mid n$). Let q be some index of s^1 such that $s^1_q = 1$.

If d is the period of $s^\wedge$, then $\chi_d(s^\wedge, q)$ has period 1. Since $m = dk$, $\chi_m(s^\wedge, q)$ is a substring of $\chi_d(s^\wedge, q)$ and hence must also have period 1. However, in **Case 1** we considered $\chi_m(s^\wedge, q)$ and showed that it must have period n, and not

1. This is a contradiction, and hence $s^{\wedge}$ does not have period d where $d \mid m$. Similarly, $s^{\wedge}$ does not have period d where $d \mid n$.

Case 3: $d = m_\alpha n_\alpha$ where $m_\alpha m_\beta = m$, $n_\alpha n_\beta = n$. If $m_\beta = n_\alpha = 1$, or if $m_\alpha = n_\beta = 1$, we have **Case 1**.

If $m_\alpha \neq 1$ and $m_\beta \neq 1$, while $n_\alpha = 1$, or if $n_\alpha \neq 1$ and $n_\beta \neq 1$, while $m_\alpha = 1$, we have **Case 2**.

We now consider the case where neither m_α, n_α nor n_β are equal to 1.

Assume that $s^{\wedge}$ has period $m_\alpha n_\alpha$. We first seek to establish that for any index of the form rm_α, we have $s^{1\prime}_{rm_\alpha} = 1$, in order to reason about the values of $s^{2\prime}$ at indices of the form rm_α.

Since $s^{1\prime}$ has period m and $s^{2\prime}$ has period n, there exists some q_0 so that $s^{\wedge}_{q_0} = s^{1\prime}_{q_0} \wedge s^{2\prime}_{q_0} = 1$, and hence $s^{1\prime}_{q_0} = s^{2\prime}_{q_0} = 1$. We can shift $s^{1\prime}$ and $s^{2\prime}$ by the same amount without affecting the period of the resulting $s^{\wedge}$, and so we may assume, without loss of generality, that $q_0 = 0$.

Now, by assumption, $\chi_{m_\alpha n_\alpha}(s^{\wedge}, q)$ has period 1, and specifically, $\chi_{m_\alpha n_\alpha}(s^{\wedge}, 0) = 1^{m_\beta n_\beta}$. Then, by Lemma 33,

$$\chi_{m_\alpha n_\alpha}(s^{1\prime}, 0) = \chi_{m_\alpha n_\alpha}(s^{2\prime}, 0) = 1^{m_\beta n_\beta} \ .$$

Furthermore, by Lemma 32, $\chi_{m_\alpha n_\alpha}(s^{1\prime}, 0) = (t^1)^{n_\beta}$, where t^1 is a n_α-displacement of $\chi_{m_\alpha}(s^1, 0)$. Consequently, since $\chi_{m_\alpha n_\alpha}(s^{1\prime}, 0) = 1^{m_\beta n_\beta}$ and consists only of the elements of $\chi_{m_\alpha}(s^1, 0)$, it follows that $\chi_{m_\alpha}(s^1, 0)$ consists entirely of 1's, and hence we have

$$s^1_0 = s^1_{m_\alpha} = s^1_{2m_\alpha} = \cdots = s^1_{(m_\beta - 1)m_\alpha} = 1 \ .$$

Now, $s^{1\prime} = (s^1)^n$, and hence

$$s^{1\prime}_0 = s^{1\prime}_{m_\alpha} = s^{1\prime}_{2m_\alpha} = \cdots = s^{1\prime}_{(nm_\beta - 1)m_\alpha} = 1 \ .$$

Therefore, any q-slice of $s^{1\prime}$ with respect to $m_\alpha n_\alpha$ where $q = rm_\alpha$ is a string consisting of 1's:

$$\begin{aligned} \chi_{m_\alpha n_\alpha}(s^{1\prime}, rm_\alpha) &= 1^{m_\beta n_\beta} \quad \text{for } 0 \leq rm_\alpha \leq m_\alpha n_\alpha - 1 \\ \therefore \chi_{m_\alpha n_\alpha}(s^{1\prime}, rm_\alpha) &= 1^{m_\beta n_\beta} \quad \text{for } 0 \leq r \leq n_\alpha - 1 \ . \end{aligned}$$

By assumption $\chi_{m_\alpha n_\alpha}(s^{\wedge}, rm_\alpha)$ has period 1, and hence from Lemma 33 it follows that $\chi_{m_\alpha n_\alpha}(s^{2\prime}, rm_\alpha)$ must have period 1 for $0 \leq r \leq n_\alpha - 1$.

But by Lemma 32, $\chi_{m_\alpha n_\alpha}(s^{2\prime}, rm_\alpha) = (t^2)^{m_\beta}$, where t^2 is an m_α-displacement of $\chi_{n_\alpha}(s^2, rm_\alpha)$,. By Lemma 25, t^2 has period 1, and so by Lemma 31, $\chi_{n_\alpha}(s^2, rm_\alpha)$ has period 1. Since by Lemma 26, $\chi_{n_\alpha}(s^2, rm_\alpha) = \chi_{n_\alpha}(s^2, rm_\alpha \bmod n_\alpha)$, it follows that $\chi_{n_\alpha}(s^2, rm_\alpha \bmod n_\alpha)$ must also have period 1. Recall that $0 \leq r \leq n_\alpha - 1$.

Now, $\gcd(m_\alpha, n_\alpha) = 1$ and so the following are unique:

$$\begin{aligned} 0, \\ m_\alpha \bmod n_\alpha, \\ 2m_\alpha \bmod n_\alpha, \\ \vdots \\ (n_\alpha - 2)m_\alpha \bmod n_\alpha \\ \text{and } (n_\alpha - 1)m_\alpha \bmod n_\alpha. \end{aligned}$$

Since there are n_α unique values of $rm_\alpha \bmod n_\alpha$, it follows that $\chi_{n_\alpha}(s^2, q)$ has period 1 for any $0 \leq q \leq n_\alpha - 1$, and hence by Lemma 26, for any q. But then, by Lemma 28, s^2 has period p such that $p \mid n_\alpha$. This is a contradiction, and therefore $s^\wedge$ does not have period $m_\alpha n_\alpha$.

Similarly, if neither n_α, m_α nor m_β are equal to 1, $s^\wedge$ does not have period $m_\alpha n_\alpha$.

Case 4: $d = mn$ This is the only remaining case, hence $s^\wedge$ has period mn.

□

Example 27. *Let s^1 and s^2 be two primitive strings of length $m = 15$ and $n = 14$ respectively, with $s^1_0 = s^2_0 = 1$. Let $m_\alpha = 3$, $m_\beta = 5$, $n_\alpha = 7$ and $n_\beta = 2$. Furthermore, let $s^{1\prime} = (s^1)^n$ and $s^{2\prime} = (s^2)^m$, so that both $s^{1\prime}$ and $s^{2\prime}$ have length $mn = 210$.*

Now, let us suppose that $s^\wedge = s^{1\prime} \wedge s^{2\prime}$ has period $m_\alpha n_\alpha = 21$. By Lemma 28, it follows that $\chi_{21}(s^\wedge, 0) = 1^{10}$. Now, $s^1_0 = s^2_0 = 1$, and so by Lemma 33, $\chi_{21}(s^{1\prime}, 0) = \chi_{21}(s^{2\prime}, 0) = 1^{10}$.

However, by Lemma 32, $\chi_{21}(s^{1\prime}, 0) = (t^1)^2$, where t^1 is a 7-displacement of $\chi_3(s^1, 0)$. Now, $\chi_3(s^1, 0) = s^1_0 s^1_3 s^1_6 s^1_9 s^1_{12}$. A 7-displacement thereof is therefore $s^1_0 s^1_6 s^1_{12} s^1_3 s^1_9$. Consequently, $\chi_{21}(s^{1\prime}, 0) = (s^1_0 s^1_6 s^1_{12} s^1_3 s^1_9)^2 = 1^{10}$. Hence, $s^1_0 = s^1_3 = s^1_6 = s^1_9 = s^1_{12} = 1$.

Since $s^{1\prime} = (s^1)^n$, it follows that

$$s^{1\prime}_0 = s^{1\prime}_3 = s^{1\prime}_6 = \cdots = s^{1\prime}_{207} .$$

Therefore, specifically for $0 \leq r < 7$*, it follows that* $\chi_{21}(s^{1\prime}, 3r) = 1^{10}$*. Hence, by Lemma 33,* $\chi_{21}(s^{2\prime}, 3r)$ *has period 1 for* $0 \leq r < 7$*.*

However, by Lemma 32, $\chi_{21}(s^{2\prime}, 3r) = (t^2)^5$*, where* t^2 *is a* 3*-displacement of* $\chi_7(s^2, 3r)$*. By Lemma 25, the period of* $\chi_{21}(s^{2\prime}, 3r)$ *is equal to the period of* t^2*, which is equal to the period of* $\chi_7(s^2, 3r)$ *by Lemma 31, and hence equal to the period of* $\chi_7(s^2, 3r \bmod 7)$ *by Lemma 26.*

Recall that $0 \leq r < 7$*, and so we have*

$$\begin{aligned} 0 \cdot 3 \bmod 7 &= 0, \\ 1 \cdot 3 \bmod 7 &= 3, \\ 2 \cdot 3 \bmod 7 &= 6, \\ 3 \cdot 3 \bmod 7 &= 2, \\ 4 \cdot 3 \bmod 7 &= 5, \\ 5 \cdot 3 \bmod 7 &= 1, \\ 6 \cdot 3 \bmod 7 &= 4\ . \end{aligned}$$

Hence, $\chi_7(s^2, 3r \bmod 7)$ *has period 1 for* 7 *unique values of* $q = 3r \bmod 7$*, and hence for any* $0 \leq q < 7$*, and consequently by Lemma 26 for any* q*. Then, by Lemma 28,* s^2 *has some period* p *such that* $p \mid 7$*. But* s^2 *has period* 14*, and hence we have a contradiction.* □

Theorem 35. *Let* s^1 *and* s^2 *be two primitive binary strings of length* m *and* n*, where* $\gcd(m, n) = 1$ *and hence* $\operatorname{lcm}(m, n) = mn$*. Let* $s^{1\prime} = (s^1)^n$ *and* $s^{2\prime} = (s^2)^m$*, so that* s'_1 *and* s'_2 *are strings of length* mn*. Now, let* $s^\vee = s^{1\prime} \vee s^{2\prime}$*, i.e.* $s^\vee$ *is the result of performing a bitwise OR operation on* $s^{1\prime}$ *and* $s^{2\prime}$*. Then* $s^\vee$ *is a primitive string.*

Proof. Since the theorem concerns the bitwise OR operation between two strings, we are specifically interested in those positions where a 0 occurs in both strings, since these are the only pairs of bits that will result in a 0 in $s^\vee$. In the proof for Theorem 34, we limited the selection of q-slices to those where 1's occur in $s^{1\prime}$ in order to reason about 1's occurring in $s^{2\prime}$. A proof for the bitwise OR case would be identical, except that we would limit the selection of q-slices to those where 0's occur in $s^{1\prime}$ in order to reason about 0's occurring in $s^{2\prime}$. In particular, the proof relies similarly on Lemma 33, but simply invokes a different case. □

Theorem 36. *Let* s^1 *and* s^2 *be two primitive binary strings of length* m *and* n*, where* $\gcd(m, n) = 1$ *and hence* $\operatorname{lcm}(m, n) = mn$*. Let* $s^{1\prime} = (s^1)^n$ *and* $s^{2\prime} = (s^2)^m$*, so that* s'_1 *and* s'_2 *are strings of length* mn*. Now, let* $s^- = s^{1\prime} - s^{2\prime}$*, i.e.* s^- *is the*

result of performing a bitwise DIFF operation on $s^{1\prime}$ and $s^{2\prime}$. Then s^{-} is a primitive string.

Proof. Since the theorem concerns the bitwise DIFF operation between two strings, we are specifically interested in those positions where a 1 occurs in $s^{1\prime}$ and a 0 occurs in $s^{2\prime}$, since these are the only pairs of bits that will result in a 1 in s^{-}. As in the case of Theorem 35, the proof is similar to that of Theorem 34, relying on Lemma 33 but invoking a different case. □

Having established the previous three theorems, we are now in a position to apply them to operations on languages represented by XNFA.

5.2 Descriptional complexity of language operations for unary XNFA: intersection, union and relative complement

Unary XDFA cycles have similar behaviour to linear feedback shift registers (LFSRs) [33], and consequently, XNFA whose characteristic polynomials are primitive polynomials or products of primitive polynomials have good equidistribution properties [17; 40]. Specifically, Wang and Compagner [40] refer to three randomness properties, **R1**, **R2** and **R3**, which were first introduced by Golomb in [10]. They show that the difference in these properties between maximal length cycles derived from primitive polynomials and those derived from products of primitive polynomials are negligible. A detailed discussion of these properties is beyond the scope of this work, but it suffices to say that they guarantee that in the applicable XDFA cycles, the number of final states are only one more than the number of non-final states, and they limit the length of so-called runs of final and non-final states, leading to a uniform distribution of final states in the cycles.

As we have noted, the language accepted by an XNFA can be represented by a binary string s which corresponds to the cycle of the equivalent minimal XDFA. Since the XDFA is minimal, s must be a primitive string, and in the case presented here, it has good equidistribution properties. We use these characteristics to show that $m+n$, a lower bound on the descriptional complexity of the intersection, union and relative complement language operations for unary XNFA, cannot be reached for all m and n.

First, we show that $(2^m-1)(2^n-1)$ is the upper bound on the operation of intersection applied to languages represented minimally by two XNFA with m and n states, respectively.

Theorem 37. *Let $N^1 = (Q^1, \Sigma, \delta^1, Q_0^1, F^1)$ and $N^2 = (Q^2, \Sigma, \delta^2, Q_0^2, F^2)$ be an m-state and an n-state unary XNFA, respectively, and let $2^m - 1$ and $2^n - 1$ be relatively prime. Furthermore, let $Q_0^1 = Q_0^2 = F^1 = F^2 = \{q_0\}$. Let $\mathcal{L}^1$ and $\mathcal{L}^2$ be the languages recognised by N^1 and N^2, respectively. Then the minimal XDFA that accepts $\mathcal{L} = \mathcal{L}^1 \cap \mathcal{L}^2$ has $(2^m - 1)(2^n - 1)$ states.*

Proof. First, let s^1 be the binary string of length $2^m - 1$ representing $\mathcal{L}^1$, corresponding to its XDFA cycle of length $2^m - 1$, and let s^2 be the binary string of length $2^n - 1$ representing $\mathcal{L}^2$, corresponding to its XDFA cycle of length $2^n - 1$. Now, let $s^{1\prime} = (s^1)^{2^n-1}$ and let $s^{2\prime} = (s^2)^{2^m-1}$. Then $s^\wedge = s^{1\prime} \wedge s^{2\prime}$, which has length $(2^m - 1)(2^n - 1)$, would represent $\mathcal{L} = \mathcal{L}^1 \cap \mathcal{L}^2$. Note that since $(2^m - 1)$ and $(2^n - 1)$ are relatively prime, $\text{lcm}(2^m - 1, 2^n - 1) = (2^m - 1)(2^n - 1)$, and hence no larger DFA cycle for the intersection of $\mathcal{L}^1$ and $\mathcal{L}^2$ would be minimal. Since $\gcd(2^m - 1, 2^n - 1) = 1$, by Theorem 34, $s^\wedge$ is primitive, and hence a DFA cycle recognising $\mathcal{L}$ must have exactly $(2^m - 1)(2^n - 1)$ states. Hence, $(2^m - 1)(2^n - 1)$ is an upper bound. □

The following lemma allows us to conclude that for any language that requires a cycle of length $(2^m - 1)(2^n - 1)$ in its minimal XDFA, $m + n$ is a lower bound on the descriptional complexity of the equivalent XNFA representing that language.

Lemma 38. *For $m, n \geq 2$, a minimal XDFA cycle of length $(2^m - 1)(2^n - 1)$ requires at least $m + n$ states in an equivalent XNFA or ⋆-XNFA.*

Proof. Let $r = m + n - 1$. Then the largest cycle length achievable by an XNFA or ⋆-XNFA with r states is $2^r - 1$. However,

$$\begin{aligned}(2^m - 1)(2^n - 1) &= 2^{m+n} - 2^m - 2^n + 1 \\ &> 2^{m+n} - (2^m + 2^n) \\ &geq 2^{m+n} - 2^{m+n-1} \\ &= 2^{m+n-1} \\ &> 2^{m+n-1} - 1 \\ &= 2^r - 1 \ .\end{aligned}$$

Hence, at least $m + n$ XNFA states or ⋆-XNFA are necessary to achieve a cycle of length $(2^m - 1)(2^n - 1)$. □

The following theorem states that for certain XNFA with m states and n states, respectively, an XNFA with $m + n$ states is insufficient for representing the intersection of the two languages represented by the XNFA.

Theorem 39. *Let $N^1 = (Q^1, \Sigma, \delta^1, Q_0^1, F^1)$ and $N^2 = (Q^2, \Sigma, \delta^2, Q_0^2, F^2)$ be an m-state and an n-state unary XNFA, respectively, and let $2^m - 1$ and $2^n - 1$ be relatively prime. Let M_1 and M_2, their respective transition matrices, be the non-singular normal form matrices of primitive polynomials $c_m(X)$ and $c_n(X)$. Furthermore, let $Q_0^1 = Q_0^2 = F^1 = F^2 = \{q_0\}$. Let $\mathcal{L}^1$ and $\mathcal{L}^2$ be the languages recognised by N^1 and N^2, respectively. Then the smallest XNFA that accepts $\mathcal{L} = \mathcal{L}^1 \cap \mathcal{L}^2$ must have more than $m + n$ states.*

Proof. As shown in the proof of Theorem 37, the equivalent DFA that accepts $\mathcal{L}$ has $(2^m - 1)(2^n - 1)$ states. By Lemma 38, the cycle length $(2^m - 1)(2^n - 1)$ in an XDFA requires at least $m + n$ states in an equivalent XNFA.

Any XNFA with exactly $m + n$ states that leads to an equivalent XDFA that is a cycle of length $(2^m - 1)(2^n - 1)$ must have a characteristic polynomial that is reducible and has two primitive polynomial factors $c_m(X)$ and $c_n(X)$ of degree m and n respectively, by Theorem 4 on page 18. Such an XNFA is equivalent to the XNFA whose transition matrix contains two blocks that are the normal form matrices of $c_m(X)$ and $c_n(X)$, respectively. Let $c(X) = c_m(X)c_n(X)$, which is the polynomial of least degree with factors $c_m(X)$ and $c_n(X)$, and let N be an XNFA with characteristic polynomial $c(X)$ whose transition matrix is the block diagonal matrix of $c(X)$. We may assume that the initial states of N are chosen in such a way that its equivalent XDFA is the appropriate cycle of length $(2^m - 1)(2^n - 1)$.

According to [17], given any choice of final states, the cycle of length $(2^m - 1)(2^n - 1)$ in the cycle structure of $c(X)$ has good equidistribution properties, and specifically the property **R1**, which is that the number of non-final states is only one less than the number of final states [40]. We may therefore say that the language $\mathcal{L}_N$ accepted by N is equidistributed.

However, according to [17] and [40], the languages $\mathcal{L}^1$ and $\mathcal{L}^2$ are similarly equidistributed. This means that, given strings $s^{1\prime}$ and $s^{2\prime}$ and considering pairs of the form $(s_i^{1\prime}, s_i^{2\prime})$, the pairs $(0,0)$, $(0,1)$, $(1,0)$ and $(1,1)$ are equidistributed over the length of the strings. Therefore, $s^{\wedge}$, the result of the bitwise AND operation on $s^{1\prime}$ and $s^{2\prime}$, does not have the property **R1**, since occurrences of the pairs $(0,0)$, $(0,1)$, $(1,0)$ result in a 0 in $s^{\wedge}$, while only occurrences of the pair $(1,1)$ result in a 1 in $s^{\wedge}$. Hence, $\mathcal{L}_N \neq \mathcal{L}$, and consequently, no XNFA with $m + n$ states, which leads to a cycle of length $(2^m - 1)(2^n - 1)$ with good equidistribution properties, will accept $\mathcal{L}$. □

The following two theorems give upper and lower bounds for union and relative complement, respectively, which are identical to those established for intersection.

Theorem 40. *Let $N^1 = (Q^1, \Sigma, \delta^1, Q_0^1, F^1)$ and $N^2 = (Q^2, \Sigma, \delta^2, Q_0^2, F^2)$ be an m-state and an n-state unary XNFA, respectively, and let $2^m - 1$ and $2^n - 1$ be relatively prime. Let M_1 and M_2, their respective transition matrices, be the non-singular normal form matrices of primitive polynomials $c_m(X)$ and $c_n(X)$. Furthermore, let $Q_0^1 = Q_0^2 = F^1 = F^2 = \{q_0\}$. Let $\mathcal{L}^1$ and $\mathcal{L}^2$ be the languages recognised by N^1 and N^2, respectively. Then the minimal XDFA that accepts $\mathcal{L} = \mathcal{L}^1 \cup \mathcal{L}^2$ has $(2^m - 1)(2^n - 1)$ states, while the smallest XNFA requires more than $m + n$ states.*

Proof. The proof is similar to the proof for Theorem 42, except that we prove that the minimal XDFA requires $(2^m - 1)(2^n - 1)$ on the basis of Theorem 35. As for the intersection case, $\mathcal{L} = \mathcal{L}^1 \cup \mathcal{L}^2$ is not an equidistributed language, since occurrences of the pairs $(0, 1)$, $(1, 0)$ and $(1, 1)$ result in a 1 in $s^\vee$, while only occurrences of the pair $(0, 0)$ result in a 0. Hence, no XNFA with $m + n$ states, which leads to a cycle of length $(2^m - 1)(2^n - 1)$ with good equidistribution properties, will accept $\mathcal{L}$. □

Theorem 41. *Let $N^1 = (Q^1, \Sigma, \delta^1, Q_0^1, F^1)$ and $N^2 = (Q^2, \Sigma, \delta^2, Q_0^2, F^2)$ be an m-state and an n-state unary XNFA, respectively, and let $2^m - 1$ and $2^n - 1$ be relatively prime. Let M_1 and M_2, their respective transition matrices, be the non-singular normal form matrices of primitive polynomials $c_m(X)$ and $c_n(X)$. Furthermore, let $Q_0^1 = Q_0^2 = F^1 = F^2 = \{q_0\}$. Let $\mathcal{L}^1$ and $\mathcal{L}^2$ be the languages recognised by N^1 and N^2, respectively. Then the minimal XDFA that accepts $\mathcal{L} = \mathcal{L}^1 \setminus \mathcal{L}^2$ has $(2^m - 1)(2^n - 1)$ states, while the smallest XNFA requires more than $m + n$ states.*

Proof. The proof is similar to the proof for Theorem 42, except that we prove that the minimal XDFA requires $(2^m - 1)(2^n - 1)$ on the basis of Theorem 36. As for the intersection case, $\mathcal{L} = \mathcal{L}^1 \setminus \mathcal{L}^2$ is not an equidistributed language, since occurrences of the pairs $(0, 1)$, $(0, 0)$ and $(1, 1)$ result in a 0 in s^-, while only occurrences of the pair $(1, 0)$ result in a 1. Hence, no XNFA with $m + n$ states, which leads to a cycle of length $(2^m - 1)(2^n - 1)$ with good equidistribution properties, will accept $\mathcal{L}$. □

We have given an upper bound of $(2^m - 1)(2^n - 1)$ for the descriptional complexity of the operations of intersection, union and relative complement for two unary XNFA, with m and n states, respectively. We have also shown that $m+n$ states are insufficient for an XNFA to represent the intersection, union or relative complement of some pairs of languages that can be represented minimally by an m-state XNFA and an n-state XNFA, respectively.

Intersection, union and relative complement are asymmetric operations that result in non-equidistributed languages, as evidenced by the truth tables of their analogous boolean operations given in Figures 5.9, 5.10 and 5.11, respectively. On the other hand, symmetric difference is indeed a symmetric operation. Consequently,

its analogous boolean operation, XOR, has a symmetric truth table, shown in Figure 5.12. We might therefore expect that the descriptional complexity of the symmetric difference operation on languages represented by XNFA is different from the other operations. In the following section, we consider the descriptional complexity of these operations for ⋆-XNFA, and in Section 5.4 we give a tight bound for the symmetric difference operation on languages represented by XNFA by considering the relationship between XNFA and ⊕-XNFA.

$\wedge$	0	1
0	0	0
1	0	1

Figure 5.9: Truth table for AND

$\vee$	0	1
0	0	1
1	1	1

Figure 5.10: Truth table for OR

$-$	0	1
0	0	1
1	0	0

Figure 5.11: Truth table for DIFF

$\oplus$	0	1
0	0	1
1	1	0

Figure 5.12: Truth table for XOR

5.3 Descriptional complexity of language operations for unary ⋆-XNFA

Having shown $m + n$ states are insufficient to represent the intersection, union or relative complement of certain pairs of XNFA with m states and n states, respectively, we now show that the same operations have a descriptional complexity of $m + n$ for unary ⋆-XNFA, and that this is a tight bound.

Theorem 42. *Let $N^1 = (Q^1, \Sigma, \delta^1, Q_0^1, F^1)$ and $N^2 = (Q^2, \Sigma, \delta^2, Q_0^2, F^2)$ be an m-state and an n-state unary XNFA, respectively, and let $2^m - 1$ and $2^n - 1$ be relatively prime. Let M_1 and M_2, their respective transition matrices, be the non-singular normal form matrices of primitive polynomials $c_m(X)$ and $c_n(X)$. Furthermore, let $Q_0^1 = Q_0^2 = F^1 = F^2 = \{q_0\}$. Let $\mathcal{L}^1$ and $\mathcal{L}^2$ be the languages recognised by N^1 and N^2, respectively. Then the smallest ∩-XNFA that accepts $\mathcal{L}_\cap = \mathcal{L}^1 \cap \mathcal{L}^2$ has $m + n$ states.*

Proof. By Theorem 37, the minimal XDFA that accepts $\mathcal{L}$ is a cycle of $(2^m - 1)(2^n - 1)$ states. By Lemma 38, the cycle length $(2^m - 1)(2^n - 1)$ in an XDFA requires at least $m + n$ states in an equivalent ⋆-XNFA. Since ⋆-XNFA

Now, let $c(X) = c_m(X)c_n(X)$ and let $N_\cap = (Q, \Sigma, \delta, Q_0, \mathcal{F})$ be a ∩-XNFA with transition matrix M, where M the block diagonal matrix of $c(X)$, given below, with

A_1 being the $m \times m$ normal form matrix of $c_m(X)$ and A_2 being the $n \times n$ normal form matrix of $c_n(X)$, while the 0's represent zero matrices of appropriate sizes:

$$M = \left[\begin{array}{c|c} A_1 & 0 \\ \hline 0 & A_2 \end{array}\right] .$$

By Theorem 4 on page 18, the cycle structure of M contains three non-empty cycles. Two of them are the cycle of length $2^m - 1$, which contains $\{q_0\}$, and the cycle of length $2^n - 1$, which contains $\{q_m\}$. The structure of the block diagonal matrix ensures that XDFA states in the first cycle only contain XNFA states $q_0, q_1, \ldots, q_{m-1}$, while XDFA states in the second cycle contain only XNFA states $q_m, q_{m+1}, \ldots, q_{m+n-1}$. The third cycle has length $(2^m - 1)(2^n - 1)$, and contains all other XDFA states except the empty state. In fact, the block diagonal structure of M ensures that, starting in state $\{q_0, q_m\}$, the state S_i reached after reading a^i is $T_i^m \oplus T_i^n$, where T_i^m is the state reached after reading a^i, having started in state $\{q_0\}$, and T_i^n is the state reached after reading a^i, having started in $\{q_m\}$. However, since the states in the first and second cycles are disjunct, $T_i^m \oplus T_i^n = T_i^m \cup T_i^n$.

Hence, the language $\mathcal{L}^1$ is accepted if $Q_0 = F^{1\prime} = \{q_0\}$ and also if $Q_0 = \{q_0, q_m\}$ and $F^{1\prime} = \{q_0\}$, while the language $\mathcal{L}^2$ is accepted if $Q_0 = F^{2\prime} = \{q_m\}$ and also if $Q_0 = \{q_0, q_m\}$ and $F^{2\prime} = \{q_m\}$.

Clearly, if $\mathcal{F} = \{F^{1\prime}, F^{2\prime}\}$ for $N_\cap$, then $N_\cap$ is an $(m+n)$-state $\cap$-XNFA that accepts $\mathcal{L}_\cap = \mathcal{L}^1 \cap \mathcal{L}^2$. □

Theorem 43. *Let $N^1 = (Q^1, \Sigma, \delta^1, Q_0^1, F^1)$ and $N^2 = (Q^2, \Sigma, \delta^2, Q_0^2, F^2)$ be an m-state and an n-state unary XNFA, respectively, and let $2^m - 1$ and $2^n - 1$ be relatively prime. Let M_1 and M_2, their respective transition matrices, be the non-singular normal form matrices of primitive polynomials $c_m(X)$ and $c_n(X)$. Furthermore, let $Q_0^1 = Q_0^2 = F^1 = F^2 = \{q_0\}$. Let $\mathcal{L}^1$ and $\mathcal{L}^2$ be the languages recognised by N^1 and N^2, respectively. Then the smallest $\cup$-XNFA that accepts $\mathcal{L}_\cup = \mathcal{L}^1 \cup \mathcal{L}^2$ has $m+n$ states, the smallest $\setminus$-XNFA that accepts $\mathcal{L}_\setminus = \mathcal{L}^1 \setminus \mathcal{L}^2$ has $m+n$ states and the smallest $\oplus$-XNFA that accepts $\mathcal{L}_\oplus = \mathcal{L}^1 \oplus \mathcal{L}^2$ has $m+n$ states.*

Proof. We construct $N_\cup$ to be a $\cup$-XNFA, $N_\setminus$ to be a $\setminus$-XNFA and $N_\oplus$ to be a $\oplus$-XNFA similarly as for $N_\cap$ in Theorem 42. Then as for $N_\cap$, if $\mathcal{F} = \{F^{1\prime}, F^{2\prime}\}$ for $N_\cup$, $N_\setminus$ and $N_\oplus$, $N_\cup$ is an $(m+n)$-state $\cup$-XNFA that accepts $\mathcal{L}_\cup = \mathcal{L}^1 \cup \mathcal{L}^2$, $N_\setminus$ is an $(m+n)$-state $\setminus$-XNFA that accepts $\mathcal{L}_\setminus = \mathcal{L}^1 \setminus \mathcal{L}^2$, and $N_\oplus$ is an $(m+n)$-state $\oplus$-XNFA that accepts $\mathcal{L}_\oplus = \mathcal{L}^1 \oplus \mathcal{L}^2$. □

Since we have chosen m and n so that $2^m - 1$ and $2^n - 1$ are relatively prime, $(2^m - 1)(2^n - 1)$ is the largest minimal cycle to represent the intersection, union, relative complement or symmetric difference of two languages represented minimally

by an m-state XNFA and an n-state XNFA, respectively. We have shown that for any such choice of $c_m(X)$ and $c_n(X)$, there exists an $(m+n)$-state $\cap$-XNFA which accepts the intersection of the languages, an $(m+n)$-state $\cup$-XNFA which accepts the union of the languages, an $(m+n)$-state $\setminus$-XNFA which accepts the relative complement of the languages and an $(m+n)$-state $\oplus$-XNFA which accepts the symmetric difference of the languages. Hence, $m+n$ is a tight bound on all four operations.

We now give an example illustrating the comparison between the bounds for the union operation for XNFA and $\star$-XNFA. In particular, we show that $m+n$ states are insufficient to represent the language in question using an XNFA, but sufficient to represent the language using a $\star$-XNFA.

Example 28. *Let N^1 be an 2-state unary XNFA whose transition matrix is the normal form matrix of the polynomial $c_m(X) = X^2 + X + 1$ and let N^2 be an 3-state unary XNFA whose transition matrix is the normal form matrix of the polynomial $c_n(X) = X^3 + X + 1$. We name the states of the two XNFA as follows: $Q^1 = \{q_0, q_1\}$ and $Q^2 = \{q_2, q_3, q_4\}$. Let $Q^1_0 = \{q_0\}$ and $Q^2_0 = \{q_2\}$, while $F^1 = \{q_0\}$ and $F^2 = \{q_2\}$. Let $\mathcal{L}^1$ and $\mathcal{L}^2$ be the languages accepted by N^1 and N^2, respectively, and let $\mathcal{L} = \mathcal{L}^1 \cup \mathcal{L}^2$.*

Furthermore, let $N_\cup = (Q_\cup, \Sigma, \delta_\cup, Q_{0,\cup}, \mathcal{F})$, with $Q_0 = \{q_0, q_2\}$ and $\mathcal{F} = \{F^1_\cup, F^2_\cup\}$, be an $(m+n)$-state $\cup$-XNFA whose transition matrix M is the block diagonal matrix of the polynomial $c(X) = c_m(X)c_n(X)$, i.e. $c(X) = X^5 + X^4 + 1$, given below.

$$M = \left[\begin{array}{cc|ccc} 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{array}\right]$$

Let $F^1_\cup = \{q_0\}$ and $F^2_\cup = \{q_2\}$. Then the string associated with $\mathcal{L}^1$ is $s^1 = 101$, while the string associated with $\mathcal{L}^2$ is $s^2 = 1001011$. Then $s^{1\prime} = (101)^7$ and $s^{2\prime} = (1001011)^3$. Consequently, the string associated with $\mathcal{L}$ is

$$s^\vee = 101101111111111101111 \ .$$

Figures 5.13 and 5.14 show the cycles of N^1_D and N^2_D, the equivalent XDFA of N^1 and N^2, respectively. Furthermore, Figure 5.15 shows N_D, the equivalent XDFA of $N_\cup$, where final states determined by considering $F^1_\cup$ are indicated by a double border, while final states determined by considering $F^2_\cup$ are indicated by a thick border. Clearly, N_D accepts $\mathcal{L}$.

To see why $\mathcal{L}$ cannot be accepted by some XNFA N with the same cycle as $N_\cup$ if only one set of final states were possible, let N' be an XNFA found by changing the basis of M to M' so that M' is the normal form matrix of $c(X)$. The matrix A is given below, for which $M' = A^{-1}MA$:

$$A = \begin{bmatrix} 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

Since $Q'_0 = Q_0A$, we have $Q'_0 = \{q_0\}$. If it is possible to choose only one set of final states F for N so that $\mathcal{L}$ is accepted, it must be possible to choose an F' for the equivalent XNFA N', and indeed for any such XNFA found by changing basis. The XDFA cycle which is equivalent to N' is shown in Figure 5.16, and the states have been marked as final states as would be necessary to accept $\mathcal{L}$. However, it is clear that this selection of final DFA states does not correlate with any choice of some F' for N'. Specifically, we note that the cycle requires that $q_0, q_2, q_3 \in F'$, since $[q_0], [q_2], [q_3] \in F'_D$, and similarly that $q_1, q_4 \notin F'$. This would imply that the state $[q_0, q_1, q_2, q_4] \notin F'_D$, since it contains exactly two, i.e. and even number of, XNFA states that are in F', namely q_0 and q_2. However, for N'_D to accept $\mathcal{L}$, the state $[q_0, q_1, q_2, q_4]$ must be a final state, and so we have a contradiction.

Finally, we note that the XDFA cycle of length 21 which accepts $\mathcal{L}$ has 18 final states of which 11 are found directly adjacent to each other. Clearly, $\mathcal{L}$ is not equidis-

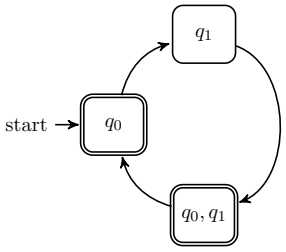


Figure 5.13: Example 28: N_D^1

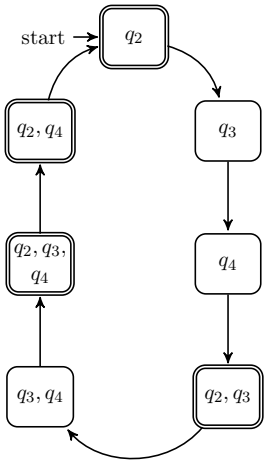


Figure 5.14: Example 28: N_D^2

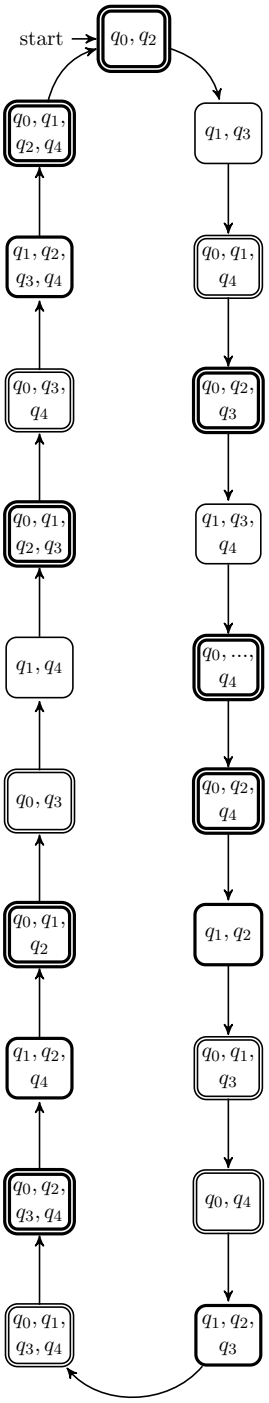


Figure 5.15: Example 28: N_D

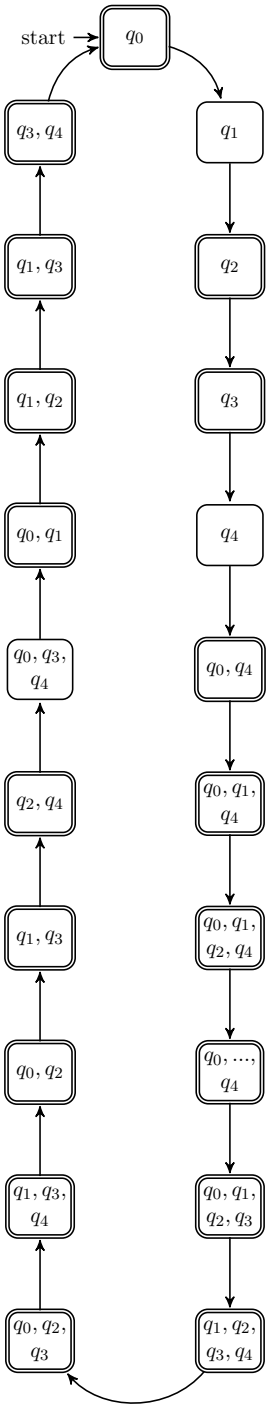


Figure 5.16: Example 28: N'_D

tributed, and hence no typical XNFA with $m+n$ states could provide a representation of $\mathcal{L}$. □

5.4 Descriptional complexity of language operations for unary XNFA: symmetric difference and complement

We have shown that the descriptional complexity of the symmetric difference operation for $\oplus$-XNFA is $m+n$ for two XNFA with m and n states respectively. The following theorem states that any unary n-state $\oplus$-XNFA is similar to some unary n-state XNFA.

Theorem 44. *For any unary n-state $\oplus$-XNFA that accepts a language $\mathcal{L}$, there exists a unary n-state XNFA that accepts $\mathcal{L}$.*

Proof. Let $N_\oplus = (Q_\oplus, \Sigma, \delta_\oplus, Q_{0,\oplus}, \mathcal{F})$ be an n-state $\oplus$-XNFA that accepts the unary language $\mathcal{L}$. Let $\Sigma = \{a\}$ and $\mathcal{F} = \{F^1, F^2, \ldots, F^r\}$ for some r. Let w be a word of length k, and let $\bar{d}$ be the vector representing the states reached after reading w, i.e. $\delta_\oplus(Q_0, w) = \bar{d}$. Recall that $w \in \mathcal{L}$ if $w \in \mathcal{L}^1 \oplus \mathcal{L}^2 \oplus \cdots \oplus \mathcal{L}^r$. Hence, the acceptance weight $\Delta_\oplus(w)$ on $N_\oplus$ is given by

$$\Delta_\oplus(w) = \bar{d}v(F^1)^T + \bar{d}v(F^2)^T + \cdots + \bar{d}v(F^r)^T \ .$$

Let $N = (Q_\oplus, \Sigma, \delta_\oplus, Q_{0,\oplus}, F)$ be an XNFA with $F = F^1 \oplus F^2 \oplus \cdots \oplus F^r$, so that

$$v(F)^T = v(F^1)^T + v(F^2)^T + \cdots + v(F^r)^T \ .$$

Then the acceptance weight $\Delta(w)$ on N is given by $\Delta(w) = \bar{d}v(F)^T$. But

$$\begin{aligned}\Delta(w) &= \bar{d}v(F)^T \\ &= \bar{d}[v(F^1)^T + v(F^2)^T + \cdots + v(F^r)^T] \\ &= \bar{d}v(F^1)^T + \bar{d}v(F^2)^T + \cdots + \bar{d}v(F^r)^T \\ &= \Delta_\oplus(w) \ .\end{aligned}$$

Hence, N is an n-state XNFA that accepts $\mathcal{L}$. □

This means that the descriptional complexity for the symmetric difference operation is equal for unary XNFA and unary ⋆-XNFA, and so the descriptional complexity of the symmetric difference operation for XNFA is $m+n$. Consequently, given two languages that can be represented succinctly by XNFA, their symmetric difference can also be represented succinctly by XNFA. Notice that, in the proof of

Theorem 44, the parity acceptance condition of XNFA is enforced by considering sets of states of XNFA as vectors over $GF(2)$. The distributive property of vectors is exploited to show that $\oplus$-XNFA are equivalent to XNFA.

The complement $\mathcal{L}^c$ of a language $\mathcal{L}$ is the relative complement $U \setminus \mathcal{L}$, where U is the universal language consisting of all strings defined over the alphabet of $\mathcal{L}$. Since the universal language is accepted by a 1-state XNFA, for a language $\mathcal{L}$ that can be represented by an n-state XNFA, by Theorem 43, its complement $\mathcal{L}^c = U \setminus \mathcal{L}$ can be represented by $n+1$-state $\setminus$-XNFA. However, $\mathcal{L}^c = U \setminus \mathcal{L} = U \oplus \mathcal{L}$, and therefore $\mathcal{L}^c$ can also be represented by an $n+1$-state $\oplus$-XNFA, and hence by an $n+1$-state XNFA, by Theorem 44. This is in fact an alternative proof of the result given in [30], which we illustrate in the following example.

Example 29. *Consider the unary language $\mathcal{L} = a^{7i+j}$, where $j \in \{0,3,5,6\}$. This language is represented by the XNFA N shown in Figure 5.17 and its equivalent XDFA N_D shown in Figure 5.18. The characteristic polynomial of N is $c(X) = X^3 + X + 1$, which leads to the XDFA cycle length of $2^3 - 1$ as seen in the figure.*

The complement of $\mathcal{L}$, namely $\mathcal{L}^c = a^{7i+j}$, where $j \in \{1,2,4\}$, is represented by the XNFA N^c shown in Figure 5.19 and its equivalent XDFA N_D shown in Figure 5.20. Notice that N^c consists of the structure of N along with a single, unconnected final state with a transition to itself. Furthermore, the initial state set consists of the two states q_0 and q_3. This means that each state in the XDFA N_D^c cycle is exactly the corresponding state of N_D with q_3, a final state, added. Consequently, due to the parity acceptance of XNFA, all final states in N become non-final in N^c, and all non-final states in N become final in N^c.

Note that N^c is the automaton whose transition matrix is the block diagonal companion matrix M^c of $c^c(X) = (X+1)c(X)$, given below:

$$M^c = \left[\begin{array}{ccc|c} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] .$$

□

The above example illustrates the argument in [30]. Notice that the extra state with a transition to itself added to N^c is exactly the XNFA that accepts the universal language U. As we have shown, this implies that, since N requires an XNFA with three states, N^c requires an XNFA with $3+1=4$ states, since the complement of a language $\mathcal{L}$ is equivalent to the symmetric difference of $\mathcal{L}$ and the universal language.

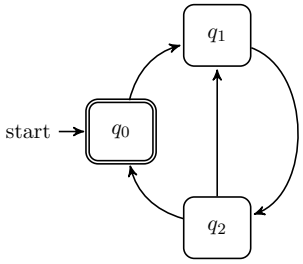


Figure 5.17: Example 29: N

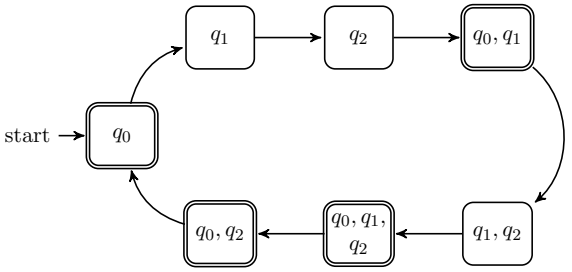


Figure 5.18: Example 29: N_D

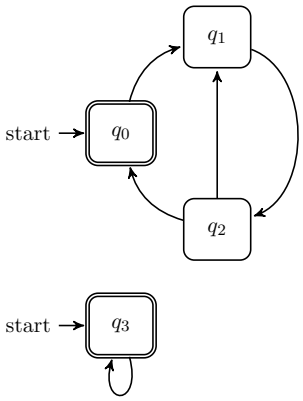


Figure 5.19: Example 29: N^c

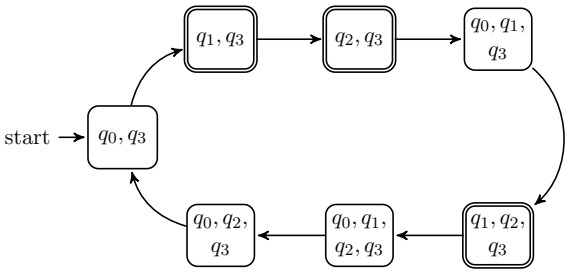


Figure 5.20: Example 29: N_D^c

We note an interesting connection here with SV-XNFA. Given a language $\mathcal{L}$ that can be represented by an n-state XNFA with a primitive characteristic polynomial, say $\phi(X)$, we know that its complement $\mathcal{L}^c$ can be represented by an $n+1$-state XNFA with characteristic polynomial $(X+1)\phi(X)$. On the other hand, we know that an $n+1$-state SV-XNFA, whose characteristic polynomial is $(X+1)\phi(X)$, can represent $\mathcal{L}$. This result makes sense in light of the consideration that an SV-XNFA requires a set of accept states as well as a set of reject states, which would correspond to the states necessary for accepting $\mathcal{L}^c$. Note, for example, that the cycle in Figure 5.20, which accepts $\mathcal{L}^c$ given $Q_0 = \{q_0, q_3\}$ and $F = \{q_0, q_3\}$, is identical to the rightmost cycle of length 7 in Figure 4.13 on page 47, which is an SV-XDFA cycle that will accept $\mathcal{L}$ given $Q_0 = \{q_0, q_3\}$, $F^a = \{q_1, q_2, q_3\}$ and $F^r = \{q_0, q_3\}$.

In proving the $m+n$ bound for the intersection, union, relative complement and symmetric difference operations of unary ⋆-XNFA, automata were constructed that did not differ much from simply running an m-state XNFA and an n-state XNFA in parallel and performing a boolean operation on the result. At first glance, this might not seem interesting. However, since all XNFA that accept the same language are similar [38], given an XNFA whose transition matrix has a block diagonal form and

whose structure is therefore not connected[3], it is always possible to find a connected XNFA that accepts the same language by performing the appropriate change of basis on the transition matrix, the initial state vector and the final state vector. In fact, given any m-state XNFA with characteristic polynomial $c_m(X)$ that accepts $\mathcal{L}_m$ and any n-state XNFA with characteristic polynomial $c_n(X)$ that accepts $\mathcal{L}_n$, the normal form matrix of $c_m(X)c_n(X)$ is guaranteed to be connected, and the appropriate choice of initial states and final states will result in a connected $\star$-XNFA that accepts the language $\mathcal{L}_m \star \mathcal{L}_n$. Furthermore, in the case of the symmetric difference operation, a connected XNFA that accepts $\mathcal{L}_m \oplus \mathcal{L}_n$ can always be found.

We give an example to illustrate the above discussion.

Example 30. *Let N^1 be a unary XNFA with transition matrix M_1, characteristic polynomial $c^1(X) = X^3 + X^2 + X + 1$ and $Q_0^1 = F^1 = \{q_0\}$ and let N^2 be a unary XNFA with transition matrix M_2, characteristic polynomial $c^2(X) = X^2 + X + 1$ and $Q_0^2 = F^2 = \{q_0\}$. We construct $N_\cap$ as a $\cap$-XNFA with transition matrix M, $Q_0 = \{q_0, q_3\}$, and $\mathcal{F} = \{F_\cap^1, F_\cap^2\}$. Let $F_\cap^1 = \{q_0\}$ and $F_\cap^2 = \{q_3\}$.*

The matrices M_1, M_2 and M are given in Figures 5.21, 5.22 and 5.23, respectively. The characteristic polynomial of M is $c(X) = c^1(X)c^2(X) = X^5 + X^3 + X^2 + 1$, and its normal form matrix M' is given in Figure 5.24.

The $\cap$-XNFA $N_\cap$ is shown in Figure 5.25, with states from $F_\cap^1$ indicated with a double border and states from $F_\cap^2$ indicated with a thick border. The automaton is not connected and has two initial states. An equivalent XNFA, $N'_\cap$, is shown in Figure 5.26, and has $F_\cap'^1 = \{q_0, q_3, q_4\}$ and $F_\cap'^2 = \{q_0, q_2, q_3\}$, also indicated with double and thick borders respectively. It is connected and has a single initial state. The respective equivalent XDFA of $N_\cap$ and $N'_\cap$, namely $N_{\cap,D}$ and $N'_{\cap,D}$, respectively, are given in Figures 5.27 and 5.28, and clearly accept the same language.

□

5.5 Towards an improved lower bound for intersection, union and relative complement for unary XNFA

We have shown that $m+n$ states are insufficient to represent the intersection, union and relative complement of some pairs of languages represented minimally by XNFA with m and n states, respectively, and hence that this lower bound is not tight. The question remains whether a better lower bound can be found.

If we consider again two languages represented minimally by XNFA with m and n states, respectively, whose equivalent XDFA have $2^m - 1$ and $2^n - 1$ states

[3]We say an automaton is connected if its structure is a connected graph.

$$M_1 = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Figure 5.21: Example 30: M_1

$$M_2 = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$$

Figure 5.22: Example 30: M_2

$$M = \left[\begin{array}{ccc|cc} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{array}\right]$$

Figure 5.23: Example 30: M

$$M' = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \end{bmatrix}$$

Figure 5.24: Example 30: M'

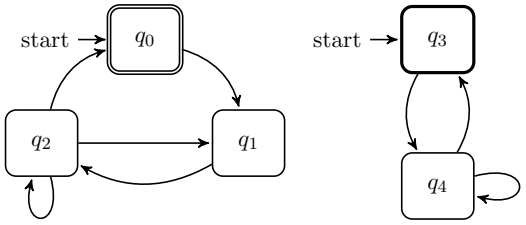


Figure 5.25: Example 30: $N_\cap$

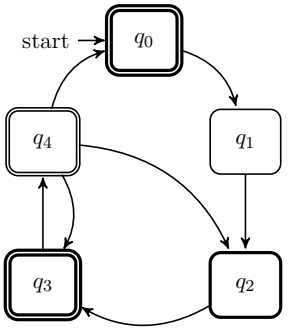


Figure 5.26: Example 30: $N'_\cap$

representing languages $\mathcal{L}_m$ and $\mathcal{L}_n$, respectively, we know that the minimal XDFA which represents the intersection, union or relative complement of $\mathcal{L}_m$ and $\mathcal{L}_n$ has $(2^m - 1)(2^n - 1)$ states (by Theorems 37, 40 and 41). While certain XNFA with $m + n$ states lead to cycles of this size, we have shown that such XNFA cannot represent the languages resulting from the intersection, union or relative complement of $\mathcal{L}_m$ and $\mathcal{L}_n$. Hence, we must consider which other XNFA lead to cycles of this size. Consequently, we must consider the four cases of polynomials given in Theorem 4 on page 18. We repeat the theorem here for convenience.

Theorem 4. [26] *Let $c(X)$ be a polynomial of degree n over $GF(2)$ that does not have X as a factor.*

- *If $c(X)$ is a primitive irreducible polynomial over $GF(2)$, then $c(X)$ has a single cycle of length $2^n - 1$, as well as the empty cycle of length 1.*
- *If $c(X)$ is an irreducible but not primitive polynomial over $GF(2)$, then $c(X)$*

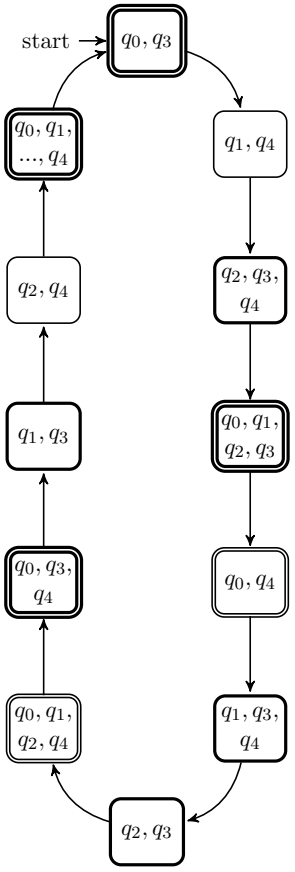


Figure 5.27: Example 30: $N_{\cap,D}$

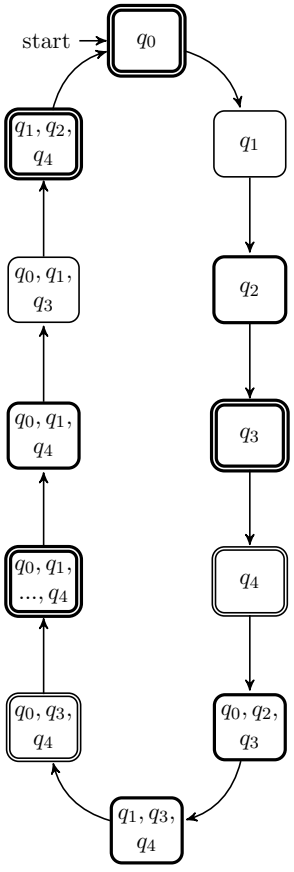


Figure 5.28: Example 30: $N'_{\cap,D}$

has $(2^n - 1)/b$ *cycles of length* b*, where* b *is a factor of* $2^n - 1$*, as well as the empty cycle of length* 1*. In fact,* b *is the period of* $c(X)$*.*

- *If* $c(X) = \phi(X)^m$ *is the power of an irreducible polynomial* $\phi(X)$ *with degree* d *over* $GF(2)$*, we let* k_i *be the period of* $\phi(X)^i$ *for each* i *such that* $1 \leq i \leq m$*. Then* $c(X)$ *has the empty cycle of length* 1*, and all the non-empty states lie on* μ_i *distinct cycles of length* k_i*, where*

$$\mu_i = (2^{di} - 2^{d(i-1)})/k_i \ , \quad \textit{for } 1 \leq i \leq m$$

- *If* $c(X)$ *is a reducible polynomial over* $GF(2)$*, consider its companion matrix. For each cycle of length* k_i *induced by block* A_i *in the companion matrix and for each cycle of length* k_j *induced by block* A_j*,* $c(X)$ *has* $\gcd(k_i, k_j)$ *cycles of length* $\operatorname{lcm}(k_i, k_j)$*.*

We can show that no primitive polynomial gives rise to a cycle of length $(2^m - 1)(2^n - 1)$.

Suppose that $(2^m-1)(2^n-1) = 2^x - 1$ for some x. Then,

$$\begin{aligned} 2^{m+n} - 2^m - 2^n + 1 &= 2^x - 1 \\ \therefore 2^{m+n} - 2^m - 2^n + 2 &= 2^x \\ \therefore 2(2^{m+n-1} - 2^{m-1} - 2^{n-1} + 1) &= 2^x \\ \therefore \log_2 2 + \log_2(2^{m+n-1} - 2^{m-1} - 2^{n-1} + 1) &= x \\ \therefore 1 + \log_2(2^{m+n-1} - 2^{m-1} - 2^{n-1} + 1) &= x \end{aligned}$$

However, $2^{m+n-1} - 2^{m-1} - 2^{n-1} + 1$ is clearly an odd number, and so its logarithm to the base 2 cannot be an integer. Consequently, x is not an integer, and hence no primitive polynomial with degree x can lead to a cycle of length $2^x - 1 = (2^m-1)(2^n-1)$.

In the case of non-primitive irreducible polynomials, all cycles have size b, where b is the period of $c(X)$ and $b \mid 2^n - 1$. We are able to give an example of a non-primitive irreducible polynomial that leads to cycles whose length can be written as $(2^m-1)(2^n-1)$. In particular, let $m = 4$ and $n = 3$. The polynomial $c(X) = X^{12} + X^8 + X^6 + X^5 + X^3 + X^2 + 1$ has period $105 = (2^4-1)(2^3-1)$, which would induce cycles of length 105. Note that $12 = 4 \times 3$; that is, we see that this polynomial which induces cycles of the required size for $m = 4$ and $n = 3$ has degree 4×3. In fact, we have verified empirically for $1 < i < 1000$, that if $(2^m-1)(2^n-1) \mid 2^i - 1$ for some m and n, then $i \geq mn$. This would suggest mn as a possible lower bound.

The third case of Theorem 4 relates to polynomials of the form $\phi(X)^m$, where $\phi(X)$ is an irreducible polynomial. Then, for each $1 \leq i \leq m$, $\phi(X)^i$ contributes cycles of length k_i, where k_i is the period of $\phi(X)^i$. However, Stone [26] gives a lemma stating that for $i > 1$, k_i is an even number, and hence the only odd-sized cycles associated with $c(X)$ are contributed by $\phi(X)$. Since $\phi(X)$ itself is irreducible, these are exactly those cycles referred to in the first and second cases of Theorem 4, which we have already discussed. Since $(2^m-1)(2^n-1)$ is an odd-sized cycle, considering powers of irreducible polynomials results in larger XNFA, without adding possibilities for cycles of length $(2^m-1)(2^n-1)$.

Finally, we must consider the remaining reducible polynomials, where cycles from different factors (of the forms given in the first three cases) induce new cycles in a pairwise fashion. We showed in Section 5.3 that the case of two primitive factors which each induce cycles of length $2^m - 1$ and $2^n - 1$, respectively, does not lead to cycles that can represent the languages resulting from intersection, union and relative complement.

For two cycles of length k_i and k_j, respectively, induced by different factors of $c(X)$, the cycle induced by the pair has length $\text{lcm}(k_i, k_j)$. The question remains whether any such pairwise combinations of cycles can lead to a new cycle of length $\text{lcm}(k_i, k_j) = (2^m - 1)(2^n - 1)$, and if so, what this implies about the degree of $c(X)$. We need only consider cycles of odd size and hence cycles contributed by primitive and non-primitive irreducible factors. Given our analysis of these cases above, it seems reasonable to suppose that no polynomial with degree less than mn could lead to a cycle of length $(2^m - 1)(2^n - 1)$. We therefore give the following conjecture.

Conjecture 45. *Given two languages represented minimally by XNFA with m and n states, respectively, mn is a lower bound on the descriptional complexity of the operations of intersection, union and relative complement for XNFA.*

The above conjecture is based solely on the possible cycle lengths associated with various polynomials. A further question, of course, is whether the languages associated with these operations can be represented by cycles derived from polynomials with degree mn. This might, as for the case of polynomials with degree $m+n$, require an investigation into the equidistribution properties of the cycles involved.

5.6 Conclusion

In this chapter we defined so-called ⋆-XNFA as an instance of generalised acceptance. We also provided a descriptional complexity upper bound of $(2^m - 1)(2^n - 1)$ on the language operations of intersection, union and relative complement for two unary languages represented minimally by an m-state XNFA and an n-state XNFA, respectively, as well as showing that the lower bound of $m + n$ is not tight. Then, we showed how the same language operations have a descriptional complexity bound, which is tight, of only $m + n$ for unary ⋆-XNFA, and hence, ⋆-XNFA are able to represent certain unary language more succinctly than XNFA. We conjectured that the descriptional complexity gap between XNFA and ⋆-XNFA for these operations is at least polynomial.

In the case of symmetric difference, we showed that the bound for ⋆-XNFA is $m + n$, and also that XNFA and ⋆-XNFA are equivalent; that is, for any n-state $\oplus$-XNFA, an equivalent n-state XNFA can be found. Hence, the descriptional complexity of the symmetric difference operation for XNFA is also $m+n$. We showed how this is consistent with the known bound of $n+1$ for the complement operation on a language represented minimally by an n-state XNFA, since the complement of a language $\mathcal{L}$ can be expressed as the symmetric difference of $\mathcal{L}$ and U, the universal language.

The proofs for showing that the lower bound on intersection, union and relative complement was not tight rested on the notion of the equidistribution of languages represented by XNFA whose characteristic polynomials are primitive polynomials or products of primitive polynomials, and the fact that these operations disturb such equidistribution properties. However, in the case of the symmetric difference set operation, the resulting equidistributed languages can indeed be represented succinctly by an XNFA due to the parity acceptance condition.

Chapter 6

Conclusion and future work

In this work, we investigated generalised acceptance conditions for symmetric difference finite state automata (XNFA). Alternative acceptance conditions, such as self-verification [6; 12; 13; 14] and acceptance conditions for various kinds of ω-automata [2; 23; 25] have been studied extensively for typical nondeterministic finite state automata (NFA), while this question had remained largely untouched for XNFA. XNFA typically exhibit cyclical behaviour, and an investigation of alternative acceptance conditions, which in some sense amounts to studying the effect of imposing or removing specific constraints upon XNFA, would contribute to a fuller understanding of various aspects of the inherently cyclical nature of XNFA. In particular, we approached these questions from a descriptional complexity point of view.

To this end, we defined generalised acceptance for XNFA, which allows multiple sets of final states. The focus of this work was to investigate the descriptional complexity of generalised acceptance conditions for XNFA with respect to two specific instances, namely self-verifying XNFA (SV-XNFA) and $\star$-XNFA. SV-XNFA have two sets of final states by definition, while in the case of $\star$-XNFA we focus specifically on the case of two sets of final states, even though more sets of final states are possible in principle. The difference between SV-XNFA and $\star$-XNFA is in the meaning associated with the sets of final states, and the nature of the constraints placed on the choice of such sets.

Before giving a formal definition for generalised acceptance XNFA (GA-XNFA), we presented an overview of the relevant characteristics of XNFA in Chapter 3. In particular, we discussed the equivalence of XNFA to weighted automata over the finite field of two elements, namely $GF(2)$, as well as the similarity between the cyclical behaviour of XNFA and that of linear feedback shift registers (LFSRs). A given XNFA is associated with a set of polynomials, which each correspond to an alphabet symbol of the XNFA. These polynomials are found by considering each

column in the transition table, and encoding its transitions as a matrix consisting of ones and zeroes. In this way, the transitions associated with each alphabet symbol for an n-state XNFA is encoded as an $n \times n$ matrix over $GF(2)$, which in turn is associated with a characteristic polynomial over $GF(2)$. The properties of these polynomials are a powerful mechanism through which to study the behaviour of XNFA.

We also noted that XNFA make use of so-called *parity acceptance*, which refers to the fact that a word is accepted by an XNFA if an odd number of paths for that word lead to final states. Indeed, this definition of acceptance is essential to maintaining the equivalence of XNFA to weighted automata over $GF(2)$, as well as similarity with LFSRs. Furthermore, the connection to linear recurrences over $GF(2)$ was introduced, which was used in the following chapter to achieve some important results.

In Chapter 4, we investigated SV-XNFA, where the two sets of final states are *accept* states and *reject* states. Here, the requirement (what we term the *SV-requirement*) for the choice of these sets is that every word be explicitly accepted or rejected by the automaton. As defined for typical self-verifying NFA (SV-NFA), this means that for any input word, at least one path must lead to a final state, and no two different paths for the same word may lead to different kinds of final states. A primary consideration in applying self-verification to XNFA was in which way the SV-requirement is to be appropriately applied in light of the parity acceptance of XNFA.

We investigated two possible applications of self-verification to XNFA, namely $GF(2)$-acceptance and disjunctive acceptance. The latter retains the intuitive requirement from SV-NFA that a state in the nondeterministic automaton belong to only one set of final states, and is therefore either an accept state or a reject state, but not both. However, we saw that this requirement removes the equivalence between SV-XNFA and weighted automata over $GF(2)$. Conversely, $GF(2)$-acceptance preserves the equivalence between XNFA and weighted automata over $GF(2)$, while introducing the notion that a state in the nondeterministic automaton may belong to both sets of final states. This is a direct consequence of parity acceptance: if an odd number of paths must lead to accept states and an even number of paths to reject states on every input (or vice versa), there is no reason to require that every path contribute to only one of the counts. However, we concluded that there must be a subset of states that belong to only one kind of final state in order to meet the SV-requirement.

The two kinds of acceptance conditions for SV-XNFA, namely $GF(2)$-acceptance or disjunctive acceptance, have the same descriptional complexity. We saw that

the state complexity of determinising SV-XNFA relates to the properties of certain characteristic polynomial(s) associated with SV-XNFA. In particular, we concluded that every polynomial associated with a given XNFA must have $X+1$ as a factor, in order to make possible a choice of accept states and reject states that meet the SV-requirement. This insight enabled us to establish a tight bound of $2^{n-1}-1$ on the state complexity of unary SV-XNFA and a lower bound of 2^{n-1} on the state complexity of SV-XNFA with larger alphabets.

In Chapter 5 we turned our focus to $\star$-XNFA, where each set of final states is associated with a language, and $\star$ represents a left-associative set operation to be performed on these languages. We investigated the descriptional complexity of such set operations on unary XNFA and contrasted the results with those for unary $\star$-XNFA: the upper bound on intersection, union and relative complement for two XNFA with m and n states, respectively, was shown to be $(2^m-1)(2^n-1)$ for XNFA, while the lower bound of $m+n$ was shown not to be tight. On the other hand, $m+n$ was shown to be a tight bound on these operations for $\star$-XNFA. Furthermore, the bound on symmetric difference was shown to be $m+n$ for both XNFA and $\star$-XNFA. In fact, we showed that $\oplus$-XNFA are equivalent to XNFA, in the sense that for any $\oplus$-XNFA an equivalent XNFA with the same number of states can always be found. Finally, the bound on complement was shown to be $n+1$ for $\star$-XNFA. We point out that complement is an instance of symmetric difference, namely for any language $\mathcal{L}$, $\mathcal{L}^c = U \setminus \mathcal{L}$, where U is the universal language over the same alphabet, and hence this confirms the result given in [30] of a descriptional complexity bound of $n+1$ on complement for XNFA.

The proofs showing that the lower bound on intersection, union and relative complement is not tight, relied on the fact that the unary languages that are most succinctly represented by XNFA are *equidistributed* languages. Informally, this means that the words of a language $\mathcal{L}$ are equidistributed over the enumeration of Σ^*: they occur as frequently as words not in $\mathcal{L}$ and are distributed more or less uniformly. The operations of intersection, union and relative complement disturb these equidistribution properties, and hence they cannot be represented succinctly by XNFA. Consequently, $\star$-XNFA, which apply these operations on the various languages associated with each set of final states, are able to represent such languages more succinctly than XNFA, and we conjectured that this gap is at least polynomial.

In the case of complement, we noted a connection with SV-XNFA. The tight upper bound for unary languages for SV-XNFA, namely $2^{n-1}-1$, is achieved for languages that can be represented with $(n-1)$-state XNFA with primitive characteristic polynomials. Hence, the same number of states is necessary to represent the complement of such languages as is necessary to represent them with SV-XNFA.

Considering that an SV-XNFA must include a set of reject states, which would essentially be equivalent to the set of accept states for the complement of the language, this result is not unexpected. By considering the polynomials that allow for succinct SV-XNFA, we saw that the presence of $X + 1$ as a factor of the characteristic polynomial(s) associated with an XNFA has certain predictable consequences for the behaviour of the XNFA. Future research might include considering the effect of the multiplicity of $X + 1$ as a factor, as well as whether any other polynomials have such a predictable effect on the behaviour of XNFA.

Several other avenues for future research present themselves immediately. In the case of SV-XNFA, it remains to prove a tight bound on the state complexity for larger alphabets. This might also lead to insights related to the conditions under which an r-ary XNFA does *not* achieve the tight state complexity bound of $2^n - 1$ for determinising [31].

Besides finding a tight bound on the operations of intersection, union and relative complement for XNFA, several other operations on XNFA remain to be investigated, such as star, concatenation, reversal and others. One may consider whether some interpretation of generalised acceptance for XNFA might produce improved bounds for these operations, as in the case for $\star$-XNFA.

Furthermore, the notion of $\star$-XNFA may itself be generalised even further. In a certain sense, $\star$-XNFA impose a simple boolean expression, namely $b_1 \star b_2 \star ... \star b_r$, on the acceptance value associated with each set of final states. One may investigate the consequences of allowing more complex expressions: presumably, languages that can be expressed in terms of a number of other languages that can be succinctly represented by XNFA, may similarly be succinctly represented by generalised $\star$-XNFA. The relation to boolean automata [18] may likewise be investigated for such XNFA.

Additionally, acceptance conditions for XNFA on infinite inputs may be studied, by, for example, investigating whether and how the acceptance conditions of Büchi automata [2] might be applied to XNFA.

In each of the cases mentioned, it may be considered to which extent a generalised understanding of acceptance may improve or shed light on various aspects of the descriptional complexity of XNFA. Finally, the notion of generalised acceptance may also be investigated for NFA and contrasted with the results for XNFA.

In summary, the following contributions were achieved in this work:

1. The introduction of generalised acceptance XNFA (GA-XNFA) as a mechanism for improving and studying descriptional complexity of XNFA under different conditions;

2. The application of self-verification to XNFA (SV-XNFA), with specific reference to two possible interpretations given parity acceptance;
3. A tight bound of $2^{n-1}-1$ on the state complexity of unary SV-XNFA;
4. An upper bound of 2^n-1 and a lower bound of 2^{n-1} on the state complexity of r-ary SV-XNFA;
5. An upper bound of $(2^m-1)(2^n-1)$ on the operations of intersection, union and relative complement on two unary XNFA with m and n states, respectively;
6. Showed that the lower bound of $m+n$ on the operations of intersection, union and relative complement on two unary XNFA with m and n states, respectively, is not tight;
7. A tight bound of $m+n$ on the symmetric difference operation on two XNFA with m and n states, respectively;
8. The introduction of $\star$-XNFA as an instance of GA-XNFA, allowing succinct representations for a greater number of languages than with XNFA;
9. A tight bound of $m+n$ on the operations of intersection, union, relative complement and symmetric difference on two unary $\star$-XNFA with m and n states, respectively;
10. Established the equivalence of $\oplus$-XNFA and XNFA;
11. A tight bound of $n+1$ on the operation of complement on a unary n-state $\star$-XNFA, with an alternative proof for the same result for XNFA as given in [30].

List of References

[1] Assent, I. and Seibert, S.: An upper bound for transforming self-verifying automata into deterministic ones. *RAIRO - Theoretical Informatics and Applications-Informatique Théorique et Applications*, vol. 41, no. 3, pp. 261–265, 2007.

[2] Büchi, J.R.: On a decision method in restricted second order arithmetic. In: Mac Lane, S. and Siefkes, D. (eds.), *The Collected Works of J. Richard Büchi*, pp. 425–435. Springer, New York, NY, 1990.

[3] Champarnaud, J.-M., Hansel, G., Paranthoën, T. and Ziadi, D.: Random generation models for NFA'S. *Journal of Automata, Languages and Combinatorics*, vol. 9, no. 2-3, pp. 203–216, 2004.

[4] Chrobak, M.: Finite automata and unary languages. *Theoretical Computer Science*, vol. 47, pp. 149–158, 1986.

[5] Dornhoff, L.L. and Hohn, F.E.: *Applied Modern Algebra*. Macmillan Publishing Co., Inc., 1978.

[6] Duris, P., Hromkovič, J., Rolim, J.D.P. and Schnitger, G.: Las Vegas versus determinism for one-way communication complexity, finite automata, and polynomial-time computations. In: Reischuk, R. and Morvan, M. (eds.), *STACS 97*, vol. 1200 of *LNCS*, pp. 117–128. Springer, Berlin, Heidelberg, 1997.

[7] Geffert, V. and Pighizzini, G.: Pairs of complementary unary languages with "balanced" nondeterministic automata. *Algorithmica*, vol. 63, no. 3, pp. 571–587, 2010.

[8] Geffert, V. and Yakaryilmaz, A.: Classical automata on promise problems. *Discrete Mathematics and Theoretical Computer Science*, vol. 17, no. 2, pp. 157–180, 2015.

[9] Goldreich, O.: On promise problems: a survey. In: Goldreich, O., Rosenberg, A. and Selman, A. (eds.), *Essays in Memory of Shimon Even*, vol. 3895 of *LNCS*, pp. 254–290. Springer, 2006.

[10] Golomb, S.W.: *Shift Register Sequences*. Aegean Park Press, 1981.

[11] Hopcroft, J.E. and Ullman, J.D.: *Introduction to Automata Theory, Languages, and Computation*. 1st edn. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1990.

[12] Hromkovic, J. and Schnitger, G.: On the power of the number of advice bits in nondeterministic computations. In: *Proceedings of the 28th ACM Symposium on Theory of Computing*, pp. 551–560. 1996.

[13] Hromkovič, J. and Schnitger, G.: On the power of Las Vegas for one-way communication complexity, OBDDs, and finite automata. *Theoretical Computer Science*, vol. 262, no. 1-2, pp. 1–24, 2001.

[14] Hromkovič, J. and Schnitger, G.: On the power of Las Vegas II: two-way finite automata. *Information and Computation*, vol. 169, pp. 284–296, 2001.

[15] Jirásek, J.S., Jirásková, G. and Szabari, A.: Operations on self-verifying finite automata. In: Beklemishev, L. and Musatov, D. (eds.), *Computer Science – Theory and Applications. CSR 2015*, vol. 9139 of *LNCS*, pp. 231–261. Springer, Cham, Switzerland, 2015.

[16] Jirásková, G. and Pighizzini, G.: Optimal simulation of self-verifying automata by deterministic automata. *Information and Computation*, vol. 209, no. 3, pp. 528–535, 2011. Special Issue: 3rd International Conference on Language and Automata Theory and Applications (LATA 2009).

[17] L'Ecuyer, P.: Tables of maximally equidistributed combined LFSR generators. *Mathematics of Computation*, vol. 68, no. 225, pp. 261–269, 1999.

[18] Leiss, E.: Succinct representation of regular languages by boolean automata. *Theoretical Computer Science*, vol. 13, pp. 323–330, 1981.

[19] Lothaire, M.: *Algebraic Combinatorics on Words*, vol. 90 of *Encyclopedia of mathematics and its applications*. Cambridge University Press, 2002.

[20] Marais, L. and Van Zijl, L.: Unary self-verifying symmetric difference automata. In: Câmpeanu, C., Manea, F. and Shallit, J. (eds.), *Proceedings of the 18th International Conference on the Descriptional Complexity of Formal Systems,*

DCFS 2016, Bucharest, Romania, July 5-8, 2016, vol. 9777 of *LNCS*, pp. 180–191. Springer, Cham, 2016.

[21] McEliece, R.J.: *Finite Fields for Scientists and Engineers*. Kluwer Academic Publishers, Norwell, MA, USA, 1987.

[22] Moreira, N., Pighizzini, G. and Reis, R.: Optimal state reductions of automata with partially specified behaviours. *Theoretical Computer Science*, vol. 658, pp. 235–245, 2017. Formal Languages and Automata: Models, Methods and Application In honour of the 70th birthday of Antonio Restivo.

[23] Rabin, M.O.: Decidability of second-order theories and automata on infinite trees. *Transactions of the American Mathematical Society*, vol. 141, pp. 1–35, 1963.

[24] Rabin, M.O. and Scott, D.: Finite automata and their decision problems. *IBM Journal of Research and Development*, vol. 3, no. 2, pp. 114–125, 1959.

[25] Safra, S. and Vardi, M.Y.: On ω-automata and temporal logic. In: *Proceedings of the Twenty-first Annual ACM Symposium on Theory of Computing*, STOC '89, pp. 127–137. ACM, New York, NY, USA, 1989.

[26] Stone, H.S.: *Discrete Mathematical Structures and their Applications*. Science Research Associates Chicago, 1973.

[27] Van der Merwe, A., Van Zijl, L. and Geldenhuys, J.: Ambiguity and structural ambiguity of symmetric difference NFAs. *Theoretical Computer Science*, vol. 537, pp. 97–104, 2014.

[28] Van der Merwe, B.: Minimal weighted automata over the Galois Field with two elements. In: Gruner, S. and Watson, B. (eds.), *Formal Aspects of Computing: Essays Dedicated to Derrick Kourie on the Occasion of his 65th Birthday*. Shaker Verlag GmbH, Aachen, Germany, 2013.

[29] Van der Merwe, B., Tamm, H. and Van Zijl, L.: Minimal DFA for symmetric difference NFA. In: Kutrib, M., Moreira, N. and Reis, R. (eds.), *Proceedings of the 14th International Conference on the Descriptional Complexity of Formal Systems, DCFS 2012, Braga, Portugal, July 23-25, 2012*, vol. 7386 of *LNCS*, pp. 307–318. Springer, Berlin, Heidelberg, 2012.

[30] Van der Merwe, B., Van Zijl, L. and Geldenhuys, J.: Ambiguity of unary symmetric difference NFAs. In: Cerone, A. and Pihlajasaari, P. (eds.), *Proceedings of the 8th International Colloquium on Theoretical Aspects of Computing (ICTAC 2011)*, vol. 6916 of *LNCS*, pp. 256–266. Springer, 2011.

[31] Van Zijl, L.: *Generalized nondeterminism and the succinct representation of regular languages*. Ph.D. thesis, Stellenbosch University, 1997. Available at `http://www.cs.sun.ac.za/~lvzijl/publications/boek.ps.gz`.

[32] Van Zijl, L.: Nondeterminism and succinctly representable regular languages. In: *Proceedings of the 2002 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists*, SAICSIT '02, pp. 212–223. Republic of South Africa, 2002.

[33] Van Zijl, L.: Random number generation with $\oplus$-NFAs. In: Watson, B.W. and Wood, D. (eds.), *Implementation and Application of Automata: 6th International Conference, CIAA 2001 Pretoria, South Africa, July 23–25, 2001 Revised Papers*, vol. 2494 of *LNCS*, pp. 263–273. Springer, Berlin, Heidelberg, 2002.

[34] Van Zijl, L.: On binary $\oplus$-NFAs and succinct descriptions of regular languages. *Theoretical Computer Science*, vol. 328, pp. 161–170, 2004.

[35] Van Zijl, L.: Magic numbers for symmetric difference NFAs. *International Journal of Foundations of Computer Science*, vol. 16, no. 05, pp. 1027–1038, 2005.

[36] Van Zijl, L. and Geldenhuys, J.: Descriptional complexity of ambiguity in symmetric difference NFAs. *Journal of Universal Computer Science*, vol. 17, no. 6, pp. 874–890, 2011.

[37] Van Zijl, L., Muller, G. and Daciuk, J.: Minimization of unary symmetric difference NFAs. *South African Computer Journal*, vol. 2005, no. 34, pp. 69–75, 2005.

[38] Vuillemin, J. and Gama, N.: Compact normal form for regular languages as Xor automata. In: Maneth, S. (ed.), *Proceedings of the 14th International Conference on the Implementation and Application of Automata (CIAA 2009), Sydney, Australia, July 14-17, 2009*, vol. 5642 of *LNCS*, pp. 24–33. Springer, Berlin, Heidelberg, 2009.

[39] Vuillemin, J. and Gama, N.: Efficient Equivalence and Minimization for Non Deterministic Xor Automata. Research report, INRIA, May 2010. Available at `https://hal.inria.fr/inria-00487031/file/MXA.pdf`.

[40] Wang, D.K. and Compagner, A.: On the use of reducible polynomials as random number generators. *Mathematics of Computation*, vol. 60, pp. 363–374, 1993.

Appendix A

Examples for Chapter 3

This appendix contains some examples to further illustrate various concepts discussed in Chapter 3. References are given to the example or section for which they provide more detail.

Example 31. *In Example 5 on page 19, the cycle structures of four different kinds of polynomials are discussed, namely primitive polynomials, non-primitive irreducible polynomials, powers of irreducible polynomials and other reducible polynomials. In Example 6 on page 21, an example of the latter is given for the normal form matrix of* $c_d(X) = (X+1)(X^3+X+1)$. *Here, we give the cycle structures associated with the normal form matrices of the primitive polynomial* $c_a(X) = X^4+X+1$ *in Figure A.1, the non-primitive irreducible polynomial* $c_b(X) = X^4+X^3+X^2+X+1$ *in Figure A.2, and* $c_c(X) = (X^2+X+1)^2$, *a power of an irreducible (primitive) polynomial, in Figure A.3.* □

Example 32. *In Section 3.3 on page 22, we mention that small structural changes to an XNFA may have profound behavioural consequences. We illustrate this by way of example.*

Compare the structures of the XNFA N_a *and* N_b *whose transition matrices are the normal form matrices of* $c_a(X) = X^4+X+1$ *and* $c_b(X) = X^4+X^2+X+1$,

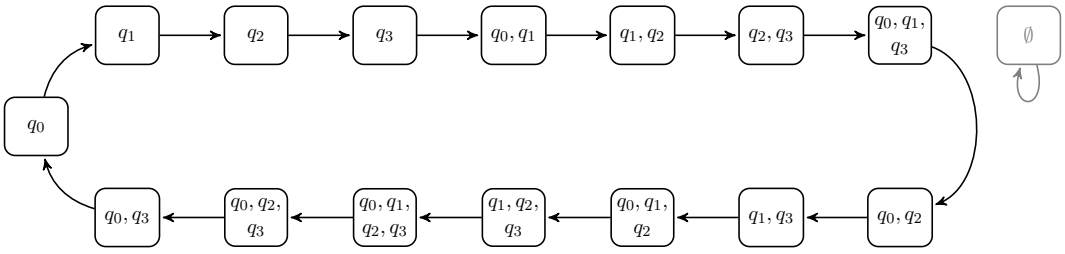


Figure A.1: Primitive polynomial: $c_a(X) = X^4+X+1$

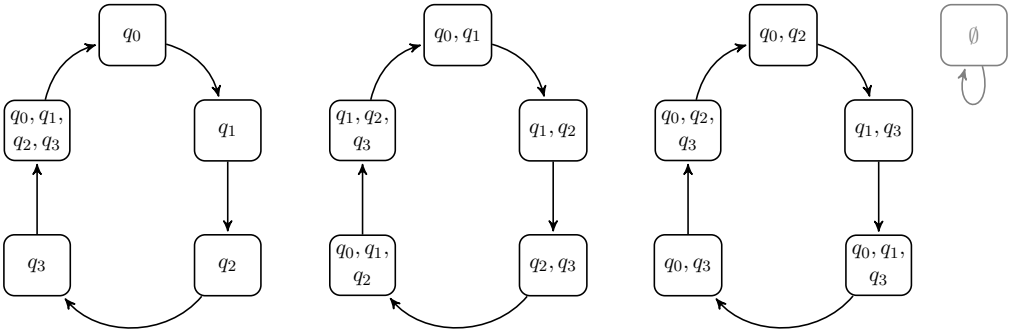


Figure A.2: Non-primitive irreducible polynomial: $c_b(X) = X^4 + X^3 + X^2 + X + 1$

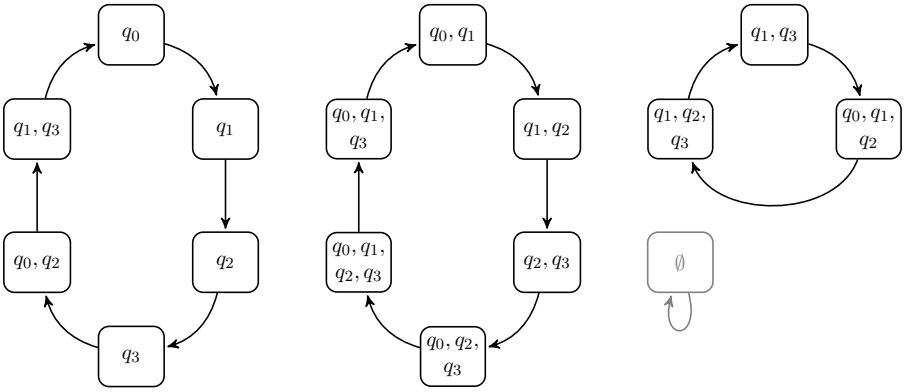


Figure A.3: Power of primitive a polynomial: $c_c(X) = (X^2 + X + 1)^2$

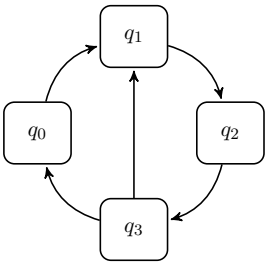


Figure A.4: Structure of N_a

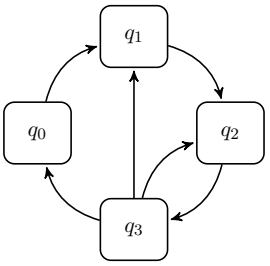


Figure A.5: Structure of N_b

given in Figures A.4 and A.5, respectively. The only difference is the addition of the transition from q_3 to q_2 in N_b. However, compare the structures of their equivalent XDFA, $N_{a,D}$ and $N_{b,D}$, shown in Figures A.6 and A.7, respectively. Since $c_a(X)$ is a primitive polynomial, $N_{a,D}$ consists of a single cycle containing all 15 non-empty states, irrespective of the choice of initial states. On the other hand, $c_b(X)$ is a reducible polynomial, and therefore, depending on the choice of initial states, $N_{b,D}$ may consist of one of two cycles of length 7 or a cycle of length 1. □

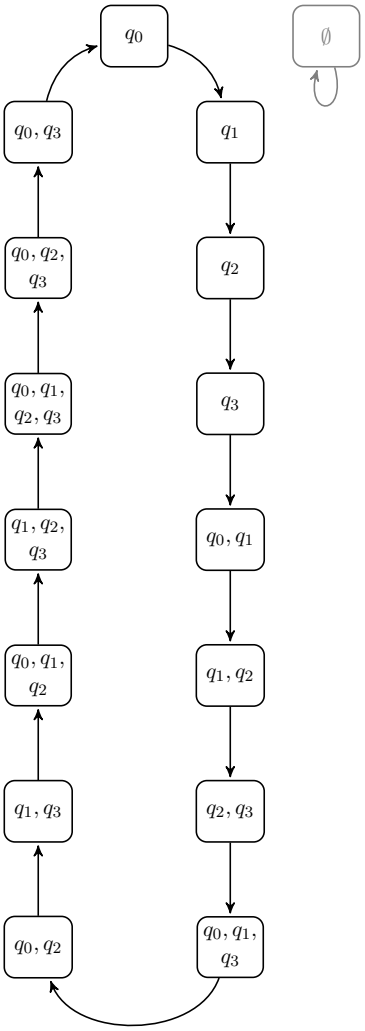


Figure A.6: Structure of $N_{a,D}$

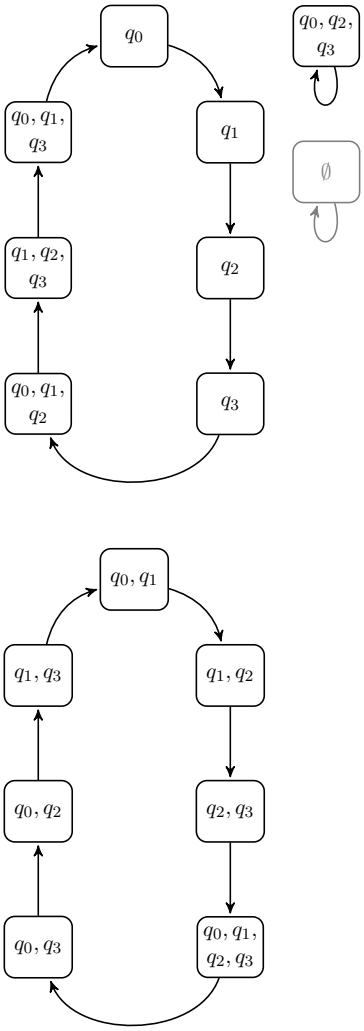


Figure A.7: Structure of $N_{b,D}$